VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

KONFIGURACE BEZDRÁTOVÉ SENZOROVÉ SÍTĚ

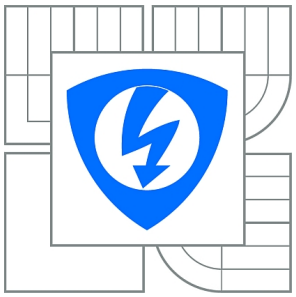
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE TOMÁŠ MINÁR
AUTHOR

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

KONFIGURACE BEZDRÁTOVÉ SENZOROVÉ SÍTĚ

WIRELESS SENSOR NETWORK CONFIGURATION

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

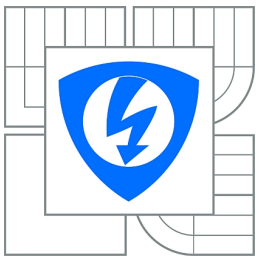
AUTOR PRÁCE — TOMÁŠ MINÁR

AUTHOR

VEDOUCÍ PRÁCE — Ing. PATRIK MORÁVEK

SUPERVISOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Tomáš Minár	***ID:***	125547
Ročník:	3	***Akademický rok:***	2011/2012

NÁZEV TÉMATU:

Konfigurace bezdrátové senzorové sítě

POKYNY PRO VYPRACOVÁNÍ:

Každá platforma bezdrátových senzorových sítí implementuje a nabízí rozdílné možnosti efektivního fungování těchro zařízení. Student se ve své práci zaměří na srovnání dostupných platforem (TinyOS, Contiki, Zigbee) z hlediska odlišné implementace konfiguračních zásad a jejich praktického vlivu na výkonnost a efektivitu sítě.

DOPORUČENÁ LITERATURA:

[1] FENG, Z., LEONIDAS, G. Wireless Sensor Networks: An Information Processing Approach. Morgan Kaufmann publishers Inc., 2004. 376 s. ISBN 1-55860-914-8.

[2] MAHALIK, NITAIGOUR P., Sensor Networks and Configuration, Springer, 2007, 510 p., ISBN 978-3-540-37364-3

Termín zadání:	6.2.2012	***Termín odevzdání:***	31.5.2012

Vedoucí práce: Ing. Patrik Morávek
Konzultanti bakalářské práce:

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Práca je venovaná bezdrôtovým senzorovým sieťam z pohľadu energetickej náročnosti operačných systémov. V práci sa rozoberajú hlavne požiadavky a zásady konfigurácie senzorových uzlov s dôrazom kladeným na ich spotrebu a efektívnosť. Dokument má slúžiť ako úvod do problematiky a zoznámiť čitateľa s dostupnými používanými štandardami a ich špecifikami. Ďalej práca popisuje tri najpoužívanejšie operačné systémy: TinyOS, Contiki a BitCloud. Praktická časť je venovaná vplyvu konfigurácie operačných systémov na spotrebu na senzorovom uzle IRIS.

KĽÚČOVÉ SLOVÁ

Bezdrôtové senzorové siete, zásady konfigurácie, IEEE 802.15.4, IRIS, ZigBee, WirelessHART, 6LoWPAN, TinyOS, Contiki, BitCloud

ABSTRACT

This bachelor's thesis is dedicated to wireless sensor networks in terms of energy consumption of operating systems. Requirements and configuration principals to assure low energy consumption and effectiveness of sensor nodes are discussed in more detail. This document should serve as an introduction to wireless sensor networks and provide a reader with information about commonly used standards, their differences and field of application. Three most widely used operating systems: TinyOS, Contiki and BitCloud are described and used in practical experiments. The practical part deals with configuration of IRIS sensor node in those operating systems. This part compares the influence of different configurations on energy consumption.

KEYWORDS

Wireless sensor networks, configuration principles, IEEE 802.15.4 IRIS, ZigBee, WirelessHART, 6LoWPAN, TinyOS, Contiki, BitCloud

MINÁR, Tomáš *Konfigurace bezdrátové senzorové sítě*: bakalárska práca. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2012. 76 s. Vedúci práce bol Ing. Patrik Morávek

PREHLÁSENIE

Prehlasujem, že som svoju bakalársku prácu na tému „Konfigurace bezdrátové senzorové sítě“ vypracoval samostatne pod vedením vedúceho bakalárskej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej bakalárskej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto bakalárskej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/nebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona č. 121/2000 Sb., o právu autorském, o právoch súvisejúcich s právom autorským a o zmeně niektorých zákonov (autorský zákon), vo znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhé, hlavy VI. diel 4 Trestného zákoníka č. 40/2009 Sb.

Brno ....

(podpis autora)

POĎAKOVANIE

Rád by som poďakoval vedúcemu bakalárskej práce Ing. Patrikovi Morávkovi, za podnetné návrhy k práci, odbornú a metodickú pomoc pri spracovaní bakalárskej práce. Ďalej by som rád poďakoval Ing. Ľubomírovi Mrázovi za odborné rady.

Brno ....

(podpis autora)

OBSAH

ZOZNAM OBRÁZKOV

ÚVOD

Technológia bezdrôtových senzorových sietí sa začala vyvíjať už v roku 1998 počínajúc projektom Smartdust, ktorého koncept sa stal základom ďalšieho výskumu a vývoja na University of California v Berkeley. Táto technológia má veľký potenciál využitia, avšak jej reálne nasadenie v praxi ešte nie je bežné a rozšírené.

Práca začína popisom senzorových sietí, ich charakteristikami a možným využitím v rôznych sférach ľudskej činnosti. Prvá kapitola taktiež popisuje architektúru a požiadavky na samotný senzorový uzol ako základnú stavebnú jednotku senzorovej siete. Druhá kapitola je venovaná predovšetkým zásadám konfigurácie senzorových uzlov pre dosiahnutie dlhého života ako senzorového uzlu, tak aj celej siete. Kapitola taktiež popisuje štandard IEEE 802.15.4, základný stavebný kameň väčšiny dnešných senzorových sietí, ktorý definuje fyzický rádiový prenos a prístup na médium. Tretia kapitola venuje pozornosť špecifikám a oblastiam použitia dnes najpoužívanejších štandardov ako ZigBee, WirelessHART a 6LoWPAN, ktoré sú založené na spomínanom štandarde IEEE 802.15.4. Praktická časť pojednáva o možnostiach aplikácie troch operačných systémov: TinyOS, Contiki a Atmel BitCloud na dostupnej platforme IRIS. V tejto časti sú rozoberané základné charakteristiky operačných systémov a hlavne ich individuálny prístup a vplyv konfigurácie na šetrenie energie senzorových uzlov. Porovnanie z tohto hľadiska je overené pomocou testovacej aplikácie s rovnakým algoritmom v troch operačných systémoch pre senzorový uzol napájaný batériami. Pri testovaní sú porovnávané rozdiely medzi uplatnením a neuplatnením mechanizmov šetrenia, prípadne nastavením rozdielnych parametrov na úrovni jedného operačného systému. Následne v závere sú operačné systémy zrovnané pri uplatňovaní všetkých dostupných mechanizmov úspory energie.

1 BEZDRÔTOVÉ SENZOROVÉ SIETE

Bezdrôtová senzorová sieť WSN pozostáva z množstva zariadení rozmiestnených v sledovanej oblasti za účelom vykonávania spoločnej úlohy. Základnou stavebnou jednotkou je senzorový uzol vybavený senzorom prípadne senzormi slúžiacimi na monitorovanie aktuálnych fyzikálnych podmienok a veličín ako teplota, tlak, vlhkosť, pohyb, intenzita svetla, hluk a mnoho iných. Ďalej umožňujú dlhodobé monitorovanie ohrozených biotopov alebo seizmickej aktivity.

Každý uzol je autonómne zariadenie zložené z mikrokontroléru, postačujúcej pamäte, radiového vysielača a zdroju energie. Pri návrhu senzorových uzlov je treba brať ohľad na energetickú nenáročnosť, veľkosť pamäte, výpočtovú a prenosovú rýchlosť, pretože sú limitujúcimi faktormi pri návrhu každého WSN systému. Celá senzorová sieť môže byť zložená z teoreticky neobmedzeného počtu uzlov riadených koordinátorom. Senzorová sieť vykonáva tri základné funkcie: zber dát získaných zo senzorov, prenos dát do centrálneho uzlu (základňovej stanice), spracovanie a vyhodnotenie dát. Zároveň prostredníctvom brány môže byť senzorová sieť prístupná z Internetu, čo by mohlo znamenať rozšírenie Internetu do fyzikálneho prostredia v prípade pokrytia rozsiahlych oblastí (celé mestá) senzorovými sieťami. [1][2]

Myšlienka pripojenia fyzických objektov do Internetu dnes označovaná ako Web of Things (sieť vecí) predstavuje rožšírenie virtuálneho webového priestoru tak, že objekty skutočného sveta (označené napríklad pomocou RFID), vstavané a inteligentné zariadenia, domáce spotrebiče a WSN sú integrované nielen do siete Internet, ale aj do Webu (teda tvoria súčasť virtuálneho priestoru). Zásadný rozdiel oproti pripojeniu zariadení do Internetu za pomoci špecializovaného softvéru a proprietárnych rozhraní je v tom, že Web of Things používa dnešné webové štandardy (napríklad: HTTP, REST, URI a iné) na pripojenie zariadení do virtuálneho sveta a sprístupnenie ich služieb. [3]

1.1 Význam a využitie

Senzorové siete nachádzajú svoje využite v rôznych sférach ľudskej činnosti. Ich nasadenie závisí od prostredia použitia (drsné prostredie, priemysel, domácnosť), požiadavkov aplikácie (vysoký tlak, extrémne teploty, radiácia), spôsobu rozostavenia senzorov (rýchle nasadenie oproti rozpracovanému návrhu určitej siete), možností napájania (batérie alebo získavanie energie z prostredia). Príklady komerčného a vojenského použitia zahŕňajú [4]:

- Monitorovanie prostredia (doprava, životné prostredie, bezpečnosť)
- Ochranu infraštruktúry (rozvodné siete, dodávka vody a plynu)

- Armádne účely (klasifikácia a zameriavanie cieľa, sledovanie bojiska)
- Zdravotníctvo (sledovanie ľudských fyziologických dát, sledovanie pacientov na diaľku, starostlivosť o staršie osoby)
- Parkovacie systém
- Monitorovanie a kontrola budov
- Kontrola výrobných procesov v priemysle (automatizácia výroby, stopovanie chemických prvkov, ochrana pred haváriou)
- Logistika (sledovanie dodávky nákladu, kontrola zásob)

1.1.1 Projekt Great Duck Island

Projekt je jedným z prvých väčších aplikácii bezdrôtovej senzorovej siete na monitorovanie prostredia vtákov hniezdiacich na ostrove Great Duck. Inžinieri z University of California v Berkeley rozmiestnili 190 senzorov do hniezd vtákov, ktoré sledujú obsadenosť hniezd na základe teplotných rozdielov. Motiváciou projektu bolo zabezpečiť dohľad nad životom vtákov bez nutnosti cestovania na ostrov. Vďaka tomuto projektu vedci skúmajú získané dáta v pohodlí svojej kancelárie, ale hlavne sa minimalizovalo rušenie života vtákov, pretože mnohokrát osobná kontrola hniezd spôsobila odchod vtákov z hniezda. [1]

1.1.2 Inteligentné mestá

Dnešné veľké mestá čelia mnohým problém týkajúcich sa kvality životného prostredia a bývania. Štúdie ukazujú, že viac ako 16000 ľudí v Španielsku zomrie na následky dýchania znečisteného vzduchu, z čoho 80 % znečistenia pochádza z dopravy. Monitorovanie hladín znečistenia ovzdušia pomocou senzorových sietí poskytuje vierohodné dáta občanom. Ako uvádza Alicia Asín (Generálny riaditeľ Libelium) vo svojom článku[6], tak už existuje kompletná platforma „inteligentných miest“ (Smart Cities) od spoločnosti Libelium. Celá senzorová sieť disponuje okrem senzorov na monitorovanie plynných zlúčenín ($NO_2, CO_2, CO, CH_4, O_3$) aj monitorovanie hluku, drobných častíc(prach), technického stavu budov a mostov. [7]

Najväčšiu výzvu predstavuje riadenie dopravy a parkovania potrebné na zníženie emisií a zabezpečenie plynulej dopravy bez zápch. Množstvo senzorov sa v doprave už používa. Senzory zakomponované do vozovky alebo umiestnené popri diaľniciach vyhodnocujú stav premávky. Kamery križovatiek dohliadajú nad dodržiavaním pravidiel cestnej premávky. Senzory v autách merajú rýchlosť a iné veličiny. Lenže tieto systémy málokedy alebo vôbec nekomunikujú a nespolupracujú medzi sebou. Ak sa tieto senzory a systémy prepoja, tak si môžu medzi sebou vymieňať informácie v reálnom čase. Takýto komplexný systém by zlepšil bezpečnosť, redukoval

zahltenie ciest alebo by pomohol ľuďom v cudzom meste nájsť miesto na parkovanie. Osobné a nákladné autá vybavené bezdrôtovým systémom sa navzájom môžu upozorniť na hroziacu kolíziu alebo iné nebezpečenstvo na ceste. Taktiež dokážu dynamicky vyhodnotiť situáciu na plánovanej trase a v prípade obmedzenia alebo nepriepustnosti optimalizovať trasu. [1]

1.1.3 Sledovanie úrovní radiácie kontaminovaných a rizikových oblastí

Pri ťažbe uránu vzniká rádioaktívny odpad najčastejšie v podobe plynov, prachu a tekutín. Kontaminované oblasti predstavujú vážnu hrozbu pre človeka a prírodu. Monitorovanie radónu v ovzduší, ktorý je zdrojom alfa a beta žiarenia, je nevyhnutné na zabránenie vzniku rakoviny u ľudí spôsobenej vdýchnutím alebo požitím rádioaktívnych častíc. Z tohto dôvodu je žiadané meranie hladiny radiácie vo vzduchu v okolí aktívnych a zatvorených uránových baní. Na zber takýchto dát je vhodné použiť senzorovú sieť. Návrh experimentálneho systému, ktorý zbiera a posiela dáta o úrovni radiácii z článku [8] používa technológiu ZigBee pre svoje senzory. Túto technológiu si autori vybrali pre svoju nízku spotrebu, cenu a hlavne pre vysokú spoľahlivosť a dostatočne veľký dosah. V tomto návrhu senzory sa stávajú aktívnymi, keď úroveň radiácie prekročí stanovenú hranicu. Z pohľadu spotreby energie je to najvýhodnejšie riešenie, pretože uzly väčšinu času „spia“ a prechod do aktívneho režimu im trvá menej ako 15 ms, vďaka čomu zariadenia rýchlo reagujú na naprogramované udalosti.

1.2 Charakteristiky bezdrôtovej senzorovej siete

V bezdrôtovej senzorovej sieti prebieha komunikácia medzi jednotlivými uzlami a dochádza k prenosu užitočných a riadiacich dát na základe komunikačných protokolov pre bezdrôtové technológie.

Jednotlivé rozmiestnenie senzorov nemusí byť vždy preddefinované, čo umožňuje ich nasadenie v neprístupnom teréne alebo pri pomoci pri katastrofách. Takéto fungovanie však vyžaduje sieťové protokoly a algoritmy schopné samo organizácie.

Senzorové siete fungujú na základe spoločného úsilia senzorových uzlov. Uzly namiesto posielania surových dát zo senzorov využívajú svoje výpočtové schopnosti na základné výpočty a posielajú spracované užitočné dáta uzlu zodpovednému za vyhodnocovanie (základňová stanica taktiež nazývaná „sink“). Realizácia týchto sietí vyžaduje bezdrôtové ad hoc [1] riešenia. Napriek existencii protokolov pre tradičné ad

[1] ad hoc - voľne vytvorená sieť bez centrálneho prvku a pevnej infraštruktúry

hoc siete (Bluetooth, WiFi) museli byť vyvinuté nové, ktoré poskytujú unikátne funkcie a spĺňajú aplikačné požiadavky senzorových sietí. Rozdiely medzi senzorovými sieťami a tradičnými ad hoc sieťami:

- počet uzlov je o niekoľko rádov väčší,
- husté rozmiestnenie senzorových uzlov,
- senzorové uzly sú náchylné na zlyhanie,
- topológia senzorovej siete sa môže často meniť,
- senzorové uzly uprednostňujú formu všesmerovej komunikácie oproti forme bod-bod,
- senzorové uzly majú limitovaný zdroj energie, výpočtový výkon a pamäť,
- snaha minimalizovať režijné náklady komunikácie.

Jedným z najdôležitejších požiadavkou na senzorové uzly je nízka spotreba energie. V najbežnejších aplikáciách bývajú napájané limitovanými zdrojmi, ktoré sa môžu vymeniť. Avšak pri rozsiahlych sieťach výmena zdrojov nie je ideálna a skôr sa uprednostňuje, aby uzly po nasadení fungovali bez údržby. Preto sa samotné protokoly senzorových sietí zameriavajú primárne na spotrebu.

V senzorových sieťach je uprednostňovaná viacskoková komunikácia voči jednoskokovej. Takýto spôsob komunikácie umožňuje prenos na väčšie vzdialenosti a nízke výkonové úrovne prenosového signálu, čo býva žiadané napríklad u tajných operácií. [9]

1.2.1 Škálovateľnosť

Počet senzorov závisí od aplikácie a v extrémnych prípadoch môže dosiahnuť miliónov. Preto pri návrhu siete a protokolov je treba myslieť na budúce rozšírenie. Dôležitým parametrom je hustota siete, ktorá môže nadobúdať hodnôt jednotiek až tisícok senzorov na oblasť s priemerom menším než 10 metrov podľa [10]. Hustota môže byť vyrátaná podľa [11] ako počet uzlov na danej ploche

$$\mu(R) = \frac{N * \pi * R^2}{A}, \tag{1.1}$$

kde N je počet rozmiestnených senzorov v oblasti o rozlohe A a R je dosah rádiového prenosu, ktorý počíta s ideálnym kruhovým šírením signálu.

1.2.2 Spoľahlivosť

Senzorové uzly môžu zlyhať z dôvodu nedostatku energie, fyzického poškodenia alebo ich komunikácia môže byť prerušená silným rušením. Takéto zlyhanie by však nemalo ovplyvniť fungovanie celej siete. V článku [12] uvádzajú, že spoľahlivosť jednotlivého

uzlu je modelovaná pomocou Poissonovho rozloženia pravdepodobnosti, že senzor nezlyhá na časovom intervale *(0,t)*. Spoľahlivosť vypočítame ako

$$R_k(t) = exp(-\lambda_k t), \tag{1.2}$$

kde λ_k je chybovosť senzoru k (považovaná za konštantu).

1.2.3 Prenosové médium

Senzorové uzly komunikujú prostredníctvom bezdrôtového média. V bezdrôtovej komunikácii sa využíva určitý rozsah elektromagnetického spektra nazývaný frekvenčné pásmo. Od výberu frekvenčného pásma závisí charakteristika šírenia (napríklad ako dobre sa vlna šíri cez prekážky) a taktiež dostupná prenosová kapacita. Väčšina dnešných bezdrôtových technológií používa nelicencované komunikačné ISM pásma pre svoju globálnu dostupnosť, široké pridelené spektrum a nepodliehajú konkrétnemu štandardu. Na druhej strane pre tieto pásma sú definované isté obmedzenia týkajúce sa vysielacieho výkonu alebo výkonovej spektrálnej hustoty. Dostupné frekvenčné pásma pre ISM aplikácie sú uvedené v tab. 1.1. V bezdrôtových senzorových sieťach sa podľa [13] odporúča použitie frekvencií z UHF rádiového pásma. Z toho najčastejšie používané sú pásma 2,4 GHz, 433 MHz pre Európu a 915 MHz pre Severnú Ameriku. [5][14][15]

Tab. 1.1: Prehľad voľných ISM pásiem

Frekvenčný rozsah	**Dostupnosť**
26,957 MHz – 27,283 MHz	
40,660 MHz – 40,700 MHz	
433,050 MHz – 434,790 MHz	Európa (Región 1)
868,000 MHz – 870,000 MHz	Európa (Región 1)
902,000 MHz – 928,000 MHz	Amerika (Región 2)
2,400 GHz – 2,500 GHz	
5,725 GHz – 5,875 GHz	
24,000 GHz – 24,250 GHz	

1.3 Architektúra uzlu

Vybudovanie senzorovej siete vyžaduje, aby uzly spĺňali špecifiká danej aplikácie, napríklad musia byť mále, lacné, energeticky efektívne, vybavené potrebnými senzormi.

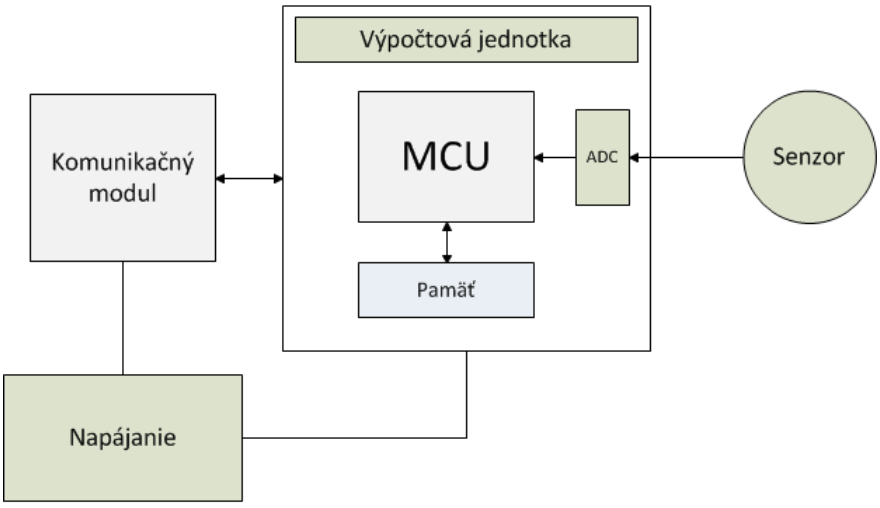
Senzorový uzol ako základný stavebný prvok siete pozostáva z napájacej, výpočtovej, senzorovej a komunikačnej jednotky. Podľa potreby môže byť rozšírený o ďalšie komponenty ako lokalizačný systém, aktuátor alebo alternatívnym zdrojom energie, ktorý využíva energiu z prostredia.

Úlohou senzorovej jednotky je snímať fyzikálne parametre prostredia. Pozostáva zo senzorov a z analógovo-číslicového prevodníku, ktorý prevádza analógový signál na digitálny, aby mohol byť ďalej spracovaný výpočtovou jednotkou.

Výpočtová jednotka je jadrom senzorového uzlu a zodpovedná za zber a vyhodnotenie dát získaných zo senzorov. Má na starosti spoluprácu s ostatnými uzlami prostredníctvom komunikačnej jednotky. Pozostáva z mikroprocesoru, pamäti, vstupných a výstupných portov, AČ prevodníkov.

Komunikačná jednotka, pozostávajúca z prijímača, vysielača a antény, zabezpečuje pripojenie uzlov do siete a výmenu informácie cez bezdrôtový komunikačný kanál. Radiová komunikácia predstavuje najdrahšiu funkciu, ktorú uzol vykonáva z pohľadu využitia energie. Preto musí byť využívaná striedmo a prispôsobená k danej aplikácii.

Napájaciu jednotku najčastejšie tvorí batéria, ktorá môže byť doplnená zariadením na získavanie energie z prostredia (napríklad svetlo, vibrácie, teplotné rozdiely, elektromagnetické žiarenie).



Obr. 1.1: Logická schéma uzlu

Požiadavky na senzorové uzly [5][14]:

- Nízka cena
  Z dôvodu potreby veľkého množstva senzorov, by tieto zariadenia mali byť navrhnuté a zostrojené za čo najmenšie finančné náklady.
- Energetická efektívnosť
  Uzly sú napájané z limitovaného zdroja, od ktorého závisí ich doba života. Preto je dôležitá efektívna správa napájania a prechod do spiaceho režimu, ak nie potrebné posielať dáta.
- Malé rozmery
  V extrémnych prípadoch požadovaná veľkosť môže byť menej ako jeden kubický centimeter (senzor súčasťou pneumatiky, ktorý monitoruje teplotu a tlak).
- Spracovanie a výpočty
  Každý uzol musí byť schopný spracovať lokálne dáta, zhodnotiť relevantnosť a premeniť ich v užitočnú informáciu. Ďalej získané informácie preniesť spoľahlivo a v prípade chyby alebo výpadku uzlu v sieti si musí nájsť novú cestu pre doručenie.
- Spoľahlivosť
  Senzorové uzly musia spoľahlivo fungovať po dĺžku svojho života, ktorá môže byť limitovaná kapacitou batérie, alebo do zničenia externými vplyvmi.
- Bezdrôtová komunikácia
  Každý uzol komunikuje priamo a výhradne so svojimi susedmi v rámci rádiového dosahu. V rámci tohto dosahu prebieha všesmerový typ komunikácie (všetky susedné uzly počujú, čo daný uzol vysiela).
- Viacskoková komunikácia
  Uzol vo väčšine prípadov nemá dosah na základňovú stanicu, preto musí byť schopný využívať susedné uzly na prenos dát mimo svoje okolie.
- Programovateľnosť
  Musí byť ľahko programovateľný, kvôli častému preprogramovanie pri vývoji komunikačných protokolov a aplikácií. Žiadaným rozšírením je možnosť zmeny firmwaru na diaľku už vo vybudovanej sieti, čo uľahčí vylepšenie stávajúcej siete.

2 ZÁSADY KONFIGURÁCIE

Energetická náročnosť jednotlivých blokov a periférií senzorového uzlu nie je rovnaká. Preto pri konfigurácií a návrhu je dôležité poznať ich spotrebu a následne si určiť priority správy napájania. Spravidla najväčšiu prioritu má správa spotreby bezdrôtovej komunikácie oproti spotrebe MCU, ktorá predstavuje okolo 30 % z celkovej spotreby. Hlavným dôvodom potreby správy energie je predĺženie života uzlu a tým pádom aj celej siete. Napríklad senzorový uzol MICA napájaný dvoma 1,5 V AA baterkami v prípade neustálej komunikácie a bez komplexnej správy spotreby vydrží 13,2 dní pri priemernej spotrebe 25 mW. Ak sa uplatní správa energie na viacerých úrovniach, tak je možné dosiahnuť doby života až 330 dní pri priemernej spotrebe 1 mW.

Správa spotreby môže byť realizovaná na úrovni platformy (hardvéru) alebo na protokolovej a aplikačnej úrovni (softvéru). V prípade hardvéru sa pre senzorové uzly vyberajú súčiastky s nízkou spotrebou. Najdôležitejšie je vybrať správny mikrokontrolér, ktorý správu napájania realizuje pomocou napájacích režimov (aktívny, nečinný, pohotovostný a odstupňované spiace režimy). Zmena pracovného výkonu mikrokontroléru sa realizuje pomocou škálovania ich pracovného taktu a napájania zdroja, čo má veľký vplyv na spotrebu; napríklad MCU MSP430C13x1 v nečinnom (idle) režime s taktom 1 MHz si odoberie 55 μA a s taktom 8 MHz 440 μA [16]. Nové rady mikrokontrolérov používajú tzv Clock gating na zníženie premenlivých energetických strát, ktorý odpojí signál hodín od jednotlivých častí integrovaného obvodu.

Priemerná spotreba pripojeného senzoru výrazne záleží od jeho typu a pohybuje sa okolo 4,7 mW u senzorového uzlu MICA. Výkonnejší uzol ITRI ZBnode používa aj CPLD obvod na správu napájania, externé pamäte SDRAM a FLASH. Pre predstavu Flash pamäť pri zápise má spotrebu maximálne 198 mW, SDRAM za behu 330 mW, CPLD obvod 40 mW a senzory použité u tohto modulu 10 mW (teplotný) a 8,5 mW (senzor intenzity dopadajúceho svetla).

Správa spotreby z pohľadu protokolov sa uplatňuje nezávisle na jednotlivých vrstvách referenčného modelu. [17]

- *Fyzická* – výber modulačnej techniky a protichybového kódovania.
- *Podvrstva MAC* – plánovanie prechodov do napájacích režimov hlavne spiaceho (S-MAC, T-MAC, Wise-MAC, techniky založené na TDMA) a zmena vysielacieho výkonu rádia (preferovanie minimálnej spotreby na prenos a použitie viacskokového prenosu na dlhšie vzdialenosti).
- *Sieťová* – znižovanie spotreby sa docieli použitím energeticky efektívnych smerovacích protokolov a správou topológie. Uprednostňované sú smerovacie protokoly, ktoré vedia rovnomerne rozložiť záťaž a spotrebu na uzly.

- *Vyššie vrstvy* – rozhodujú o tom, ktoré dáta je užitočné poslať a taktiež sa využíva agregácia dát na uzlu. Pri návrhu aplikácie je doporučené adresovať požiadavky na periférie paralelne, aby nedochádzalo k častému zapínaniu periférií.

Najviac energie sa spotrebuje v komunikačnom module, konkrétne rádiový čip CC2420 spotrebuje 19,7 mA v prijímacom režime, ale iba 1 μA vo vypnutom stave podľa [21]. Hlavné zdroje spotreby súvisiace s dátovým prenosom [2]:

- *Prenos paketov* – množstvo spotrebovanej energie je úmerné počtu prenesených paketov a času potrebnému na prenos jedného paketu.
- *Príjem paketov* – uzly niekedy prijímajú aj pakety určené iným uzlom. Každý takýto paket je spracovaný na fyzickej vrstve a k jeho zahodeniu dochádza až na podvrstve MAC. Cieľom je redukovať príjem takýchto prepočutých paketov.
- *Nečinné načúvanie* – uzol, ktorý práve nekomunikuje by nemal zbytočne načúvať na médiu, ale prejsť do úsporného alebo spiaceho režimu.
- *Dĺžka paketov* – určuje ako dlho bude prenos trvať. Na skrátenie prenosu sa môže použiť kompresia alebo sa viacej paketov vloží do jedného väčšieho, čím sa znížia režijné náklady.
- *Vzdialenosť* – má vplyv na množstvo energie potrebnej na prenos medzi uzlami.

2.1 Základné vlastnosti používaných MCU

2.1.1 Premenlivá frekvencia oscilátora

Niektoré MCU dokážu meniť svoj takt pomocou deličky na výstupe oscilátoru alebo používajú nízko frekvenčný oscilátor s kontrolovateľným násobičom. V prvom prípade systém pracuje na nižších rýchlostiach a spotrebuje menej energie, zatiaľ čo oscilátor spotrebuje maximum. Mikrokontrolér od Atmelu ATmega128L používa sedem bitovú deličku a oscilátor s frekvenciou 8 MHz. Okrem iných aplikácií je použitý v týchto senzorových uzloch: BTnode, Mica2, INDriya a Nymph.

Pri použití násobiča je možné meniť takt podľa potreby. K šetreniu energie teda dochádza vďaka použitiu oscilátora s nízkou frekvenciu často 32 kHz, napríklad MCU z rodiny MSP430 od Texas Instruments, ktorý využívajú moduly COOKIES, BEAN, Eyes, KMote, Telos, TelosB a iné. [18][19]

2.1.2 Režimy napájania

Mikrokontroléry (MCU) použité v uzloch na minimalizovanie priemerného prúdového odberu trávia čo najviac času v režime nízkej spotreby, nazývaný spiaci režim.

Keď je potrebná odozva, tak MCU prechádza do aktívneho režimu aby mohol vykonávať naprogramované funkcie.

Pracovný cyklus (duty cycle) zariadenia je časť z celkového času, ktorý zariadenie strávi v aktívnom režime. Senzorové aplikácie si kladú za cieľ dosiahnutie pracovného cyklu menšieho ako 1 %. Tohto cieľu sa dosahuje naplánovaním udalostí na určitý čas v budúcnosti a po ich vykonaní znovu prechádza do spiaceho režimu. Väčšina MCU používa asynchrónny časovač, ktorý sa spúšťa pri prechode do spiaceho režimu, kedy jadro a periférie sú odpojené. Po uplynutí časovača nastáva prerušenie a systém prechádza do aktívneho režimu. Čas prechodu do aktívneho režimu sa nazýva čas prebudenia (wakeup time). Nízky čas prebudenia umožňuje zvýšenie počtu prechodov do spiaceho režimu. Primárny faktorom vplývajúcim na čas prebudenia je ako rýchlo sa spustí a ustáli oscilátor. Niektoré architektúry nechávajú oscilátor, ktorý nie je pripojený k jadru MCU bežať celú dobu, čo umožňuje rýchly štart jadra. Takáto architektúra má nevýhodu, že počas dlhých spánkov odber energie tohto oscilátoru dominuje v spotrebe celého systému. Ale na druhej strane ak sa požívajú krátke spánkové intervaly, tak rýchle prebudenie umožňuje uspať MCU v momentoch, kedy by to inak nebolo možné. [2][16]

U MCU ATmega128L je na výber 6 úsporných režimov (Idle, ADC Noise Reduction, Power-save, Power-down, Standby a Extende Standby). Hodiny CPU v „idle“ režime môžu byť zastavené a funkcie periférií (SRAM, SPI port, systém prerušenia) zostávajú bežať. Špeciálny úsporný režim „ADC Noise Reduction“ odpája CPU a všetky I/O moduly okrem asynchrónneho časovača a AD prevodníku. V pohotovostnom režime zostáva kryštálový oscilátor spustený zatiaľ čo zvyšok zariadenia je v spiacom režime. Tento režim umožňuje rýchly nábeh do aktívneho režimu spolu s malou spotrebou.

Riešenie od TI ponúka 1 aktívny a 5 nízko-spotrebových režimov Low-power mode 0–4 (LPM), počínajúc odpojením jadra po úplné vypnutie s nezávisle zapnutými alebo vypnutými perifériami. Digitálne riadený oscilátor dovoľuje prebudenie z nízko-spotrebových režimov do aktívneho za menej než $6\,\mu s$. Prebudenie môže nastať pomocou definovaného časovača alebo po udalosti prerušenia. Po prebudení MCU vykoná inštrukcie programu prerušenia a potom sa vráti späť do LPM režimu. [16][18][19]

2.1.3 Architektúra pamäte

MSP430 používa jeden adresný priestor s RAM pamäťou pre dáta a ROM pamäťou pre hlavný program, na ktoré sa pristupuje jedným 16bitový ukazovateľom podľa Von Neumannovej architektúry.

ATMega 128L je založená na Harvardskej architektúre s pamäťovým priestorom

zvlášť pre hlavný program a zvlášť pre dáta. Má vyhradený ukazovateľ zásobníku a tri 16bitové pamäťové registre s nepriamym prístupom. [16]

2.1.4 Vypínanie hardvérových komponent

Jedným z prístupov na šetrenie energie je DPM (Dynamická správa napájania – Dynamic Power Management). Jeho úlohou je sledovať využitie podsystémov a periférií senzorového uzlu. Ak niektoré podsystémy a periférie nie sú práve používané alebo aktívne tak sa prepnú do ekonomickejšieho režimu alebo sa uspia. Napríklad v prípade, že senzorový uzol v určitých fázach fungovania nepotrebuje ukladať dáta do pamäte RAM, tak dochádza k jej odpojeniu. Ďalším príkladom je zapínanie externe pripojených senzorov vtedy, keď si program zavolá metódu čítania hodnoty zo senzoru po uplynutí časovača, ktorého hodnota závisí na konkrétnej aplikácií. V iných prípadoch zase môžeme chcieť, aby mikrokontrolér spal a k udalosti prerušenia došlo pri zmene hodnoty na senzore, ktorý zostáva stále zapnutý.

2.2 Správa napájania rádiového modulu

Rádiový modul ako nevyhnutná súčasť senzorového uzlu má najvyššie nároky na spotrebu. Pomocou MAC (Riadenie prístupu k médiu – Medium access control) protokolov určených pre WSN je možné redukovať rádiovú komunikáciu a zabezpečiť prechody do spiaceho režimu s aktívnymi pracovnými cyklami 2,5 % za minimálnej sieťovej premávky. Preto je dôležité identifikovať zdroje energetických strát (stavy kedy dochádza k plytvaniu energie).

K nečinnému načúvaniu dochádza, keď stanica počúva na médiu bez prenosu. K takémuto plytvaniu dochádza u sietí s malou premávkou a obmedzenými spiacimi režimami. Po dokončení všetkých prenosov protokoly spojovej vrstvy dovoľujú staniciam spať do začiatku ďalšieho prenosového rámca.

Ku kolízií rámcov dochádza, keď stanica pošle rámec, ktorý sa v čase prekrýva, koliduje s iným. V prípade, že sila signálu rušiaceho rámca je dostatočne veľká, tak rušiaci rámec môže znehodnotiť dáta na prijímacej strane. Hlavné dôvody prečo dochádza ku rámcovej kolízií sú: konečný čas prechodu rádiového modulu z režimu prijímania na vysielanie (250μs – 500μs) po tom, čo zistí že rádiový kanál je voľný; oneskorenie spôsobené šírením na dlhšie vzdialenosti; problematika skrytých uzlov, ktoré sú mimo dosahu posielacej stanice, ale v dosahu stanice prijímacej. Vyhýbaním sa kolízií sa zmenší počet znovu poslaných rámcov, ktoré má za následok navýšenia spotreby ako u prijímacej stanice, tak u vysielacej.

Režijné náklady protokolov si berú časť z použiteľnej šírky pásma a taktiež spotrebúvajú energiu. Preto pri návrhu protokolov sa musí myslieť na minimalizovanie

režijných nákladov (informácie na kontrolu siete, adresácia, spoľahlivé doručenie dát, určenie dostupnej cesty, a iné) za predpokladu dodržania požadovanej priepustnosti, oneskorenia a iných parametrov sieťovej komunikácie.

Na zvýšenie priepustnosti a zníženie oneskorenia sa niekedy používa technika prepočúvania správ (message overhearing), kedy stanica prijíma všetky správy a zahadzuje správy určené iným staniciam. U veľa WSN platform sa tak viacej energie spotrebuje pri prijímaní ako pri samotnom prenose. Namiesto takéhoto načúvania a na zabránenie rámcovej kolízie sa používa four-way handshake, pomocou ktorého sa zarezervuje médium pred samotným prenosom (taktiež označované ako message passing). Pritom sa používajú správy RTS (Žiadosť o posielanie – Request to Send) a CTS (Pripravený na posielanie – Clear to Send), ktoré obsahujú pole trvania oznamujúce okolitým staniciam dĺžku prenosu. Okolité stanice využívajú túto informáciu na nastavenie intervalu, ktorý strávia v spiacom režime. Po vymenení správ RTS a CTS na získanie prístupu k médiu začína prenos dávky fragmentov. Prijímacia strana po úspešnom prenose fragmentu odpovedá správou ACK. [21]

2.3 IEEE 802.15.4 štandard

Princípy fungovania komunikácie vo WSN sa líšia od tradičných bezdrôtových sietí, preto boli navrhnuté nové algoritmy a princípy komunikácie. V dnešnej dobe najpoužívanejším štandardom definujúcim fyzickú a prístupovú vrstvu k médiu je IEEE 802.15.4. Štandard bol navrhnutý asociáciou IEEE pre nízko rýchlostné bezdrôtové PAN siete v roku 2003. Štandart zahŕňa fyzickú vrstvu PHY a vrstvu riadenia prístupu k médiu MAC patriacu do spojovej vrstvy podľa referenčného modelu ISO/OSI. Až v posledných rokoch bol prijatý za vhodný komunikačný štandard pre LR-WPAN siete, častejšie označované ako PAN. Aj napriek tomu, že nebol špecificky navrhnutý pre WSN poskytuje dostatočnú flexibilitu na splnenie rôznych požiadavkou WSN sietí vďaka možnosti adekvátneho doladenia parametrov. Hlavnými vlastnosťami spĺňajúce požiadavky pre WSN sú:

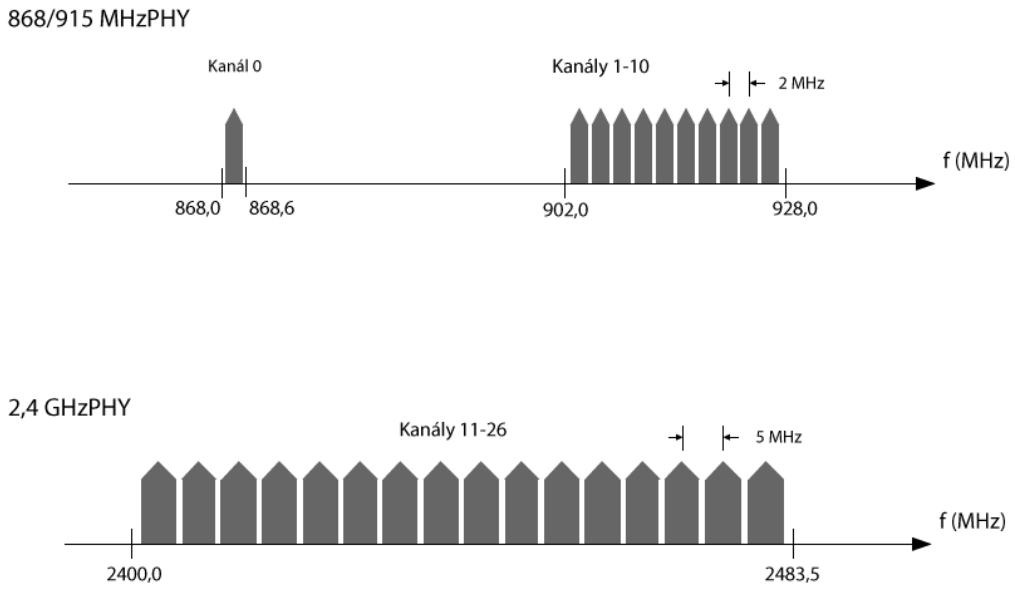
- nízka rýchlosť prenosu dát
- nízka spotreba energie
- lacné bezdrôtové spojenie.

Tento štandard predstavuje základ napríklad ZigBee špecifikácie, ktorá ďalej definuje nadradené vrstvy a tak vytvára kompletný protokolový zásobník na bezdrôtovú komunikáciu. [2]

2.3.1 Fyzická vrstva

Fyzická vrstva má na starosti samotný prenos informácie cez vzduchové médium. Protokol IEEE ponúka hneď dve riešenia PHY vrstvy. Obe riešenia používajú DSSS metódu na rozšírenie spektra pri bezdrôtovom prenose a zdieľajú rovnakú základnú štruktúru PPDU. Jedna vrstva označovaná ako 2,4 GHz PHY špecifikuje činnosť v pásme 2.4 GHz, zatiaľ čo 868/915 MHz PHY funguje v pásme 868 MHz pre Európu a 915 MHz pre Spojené štáty americké. Z dôvodu použitia iných frekvenčných pásiem je prenosová rýchlosť pre 2,4 GHz PHY 250 kb/s vďaka modulačnej technike vyššieho rádu a 868/915 MHz PHY 20 kb/s a 40 kb/s respektíve. Výber pásma závisí od použitia a charakteru WSN. Napríklad výhodou použitia pásma s nižšou prenosovou rýchlosťou je väčšia citlivosť a väčšie pokrytie územia, čo umožňuje zredukovať počet uzlov. Väčšia prenosová rýchlosť býva uprednostňovaná, keď je žiadaná väčšia priepustnosť, menšie oneskorenie alebo nižšia doba na vykonávanie strojových cyklov.

Fyzická vrstva 868/915 MHz PHY podporuje len jeden kanál v rozmedzí 868,0 a 868,6 MHz pre Európu, zatiaľ čo širšie americké pásmo používa až desať kanálov medzi 902,0 a 928,0 MHz. V pásme 2,4 GHz sa používa 16 kanálov v rozmedzí 2,4 až 2,4835 GHz s odstupom 5 MHz medzi kanálovými nosnými. Názorné rozdelenie kanálov je na obrázku 2.1.



Obr. 2.1: Kanály štandardu IEEE 802.15.4 [20]

V prostredí plnom bezdrôtových prenosov využívajúcich ISM pásma je potrebné zabezpečiť odolnosť voči rušeniu spôsobenému inými komunikujúcimi zariadeniami

alebo spotrebičmi (napríklad mikrovlnka) pomocou DSSS, monitorovania média a schopnosti preladiť sa na voľný kanál. Samotný výber kanálu neriadi fyzická vrstva, ale poskytuje vyššej vrstve funkcie ako indikácia kvality linky, výkonovej úrovne prijatého signálu a prepínanie medzi kanálmi. [20]

2.3.2 Spojová vrstva (Data Link Layer)

Podľa referenčného modelu ISO/OSI sa spojová vrstva delí na MAC a LLC podvrstvu. LLC definovaná v IEEE 802.2 štandarde je spoločná pre odvodené štandardy (IEEE 802.3, 802.11, 802.15.1 a 802.15.4), zatiaľ čo MAC vrstva je špecifická pre dané štandardy z dôvodu implementácie rozdielnych technológií fyzickej vrstvy. Hlavné funkcie, ktoré vrstva zabezpečuje sú pripojenie a odpojenie, potvrdzovanie doručených rámcov, mechanizmus prístupu na kanál, overovanie správnosti rámcov, správa časových slotov a správa vysielania signálu o prítomnosti (tzv beacon). Nadradeným vrstvám MAC podvrstva poskytuje dve základné služby: MAC dátova služba (MAC data service) a MAC správa služieb (MAC management service). V porovnaní napríklad s protokolom 802.15.1 (Bluetooth) je MAC 802.15.4 podvrstva menej komplexnejšia, čo ju robí vhodnejšou pre low-end aplikácie, kde nie je potreba množstvo funkcií.

V istých prípadoch je vyžadovaná určitá šírka pásma napríklad na dosiahnutie malého oneskorenia. Pre tieto prípady štandard ponúka režim super rámca (superframe mode). V tomto režime sieťový koordinátor (PAN koordinátor) vysiela super rámcové signály (superframe beacon) v preddefinovaných intervaloch (15 ms – 245 s). Časový úsek medzi dvoma beacon signálmi je rozdelený na 16 rovnako dlhých časových úsekov nazývaných sloty. Zariadenia patriace do siete majú povolené vysielať hocikedy v rámci časového slotu, ale musia skončiť pred vyslaním nového superframe beacon. Zariadenia za bežných podmienok súťažia o prístup ku kanálu pomocou metódy CSMA/CA, ak PAN koordinátor nepridelí časové sloty jednému zariadeniu, ktoré práve vyžaduje prenos s vyčlenenou šírkou pásma alebo malým oneskorením. Tieto špeciálne pridelené časové sloty sa nazývajú GTS (Garantované časové sloty – Guaranteed time slots) a ich dĺžka závisí na požiadavkách zariadenia. Pred použitím GTS všetky zariadenia musia dokončiť svoje prenosy. Začiatok a trvanie super rámca oznámi PAN koordinátor prostredníctvom beacon signálu ostatným zariadeniam.

V prípade siete bez signálu beacon uzly periodicky žiadajú PAN koordinátor o dáta, ktoré ak má pre ne pripravené, tak ich odošle. Perióda žiadostí sa nastavuje individuálne na danom uzle.

2.3.3 Prvky siete

IEEE 802.15.4 definuje tri typy uzlov:

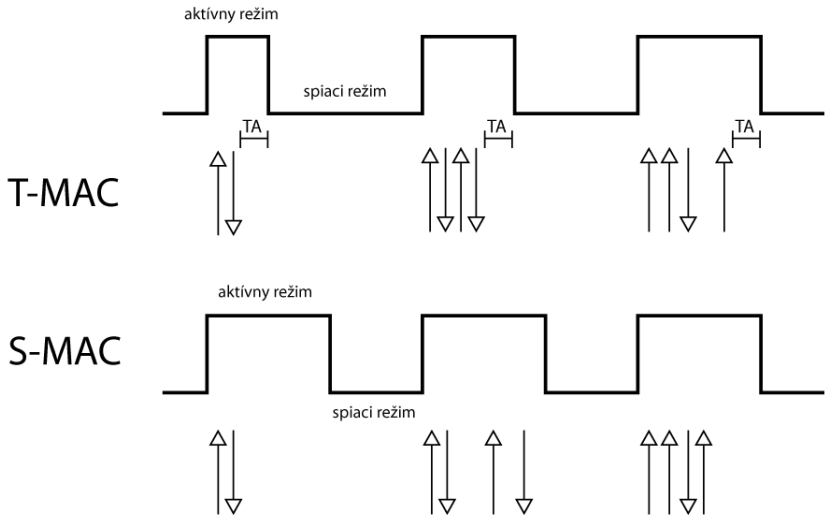
- *Sieťový* PAN *koordinátor* – zastáva funkciu hlavného operátora (pána) siete, ktorý vytvára sieť, spravuje uzly, ukladá si informácie o sieťových uzloch, preposiela správy medzi spárovanými uzlami. Je to uzol, ku ktorému sú ostatné uzly priradené. Poskytuje globálnu synchronizáciu služieb pomocou vysielania beacon rámcov, do ktorých vkladá svoj identifikátor a iné informácie.
- *Koordinátor (smerovač)* – vykonáva rovnaké funkcie ako PAN koordinátor okrem vytvárania vlastnej PAN. V podstate rozširuje pôsobnosť PAN koordinátora. Poskytuje lokálne synchronizačné služby uzlom v rámci jeho dosahu. Potrebuje stále napájanie, pretože preposiela všetky správy uzlov z jeho oblasti ďalej do siete. Zároveň môže plniť funkciu koncového zariadenia.
- *Jednoduchý uzol (koncové zariadenie)* – nevykonáva žiadne synchronizačné funkcie. Je priradený ako sluha k PAN koordinátoru prípadne koordinátoru, aby mohol byť zosynchronizovaný s ostatnými uzlami siete.

Prvé dva typy uzlov realizujú všetky funkcionality IEEE 802.15.4 protokolu pre zaistenie správy siete, preto sú nazývané ako FFD (Zariadenie s plnou funkcionalitou – Full Function Device) zariadenia. Tretí typ využíva len základné funkcie a je označovaný ako RFD (Zariadenie s obmedzenou funkcionalitou – Reduced function device). Z celého teda vyplýva, že PAN sieť musí obsahovať minimálne jedného PAN koordinátora na správu siete. [2]

2.4 Sensor MAC (S-MAC)

Hlavnou úlohou protokolu je zredukovanie spotreby energie spôsobené nečinným načúvaním (idle listening), kolíziou, prepočutím(overhearing) a režijnými nákladmi. S-MAC delí časový rámec na úsek počúvania a spania. Časový úsek počúvania sa ďalej delí na synchronizačný úsek a samotný prenos dát. Na zníženie oneskorenia a režijných nákladov si uzly medzi sebou oznamujú svoje plány prechodu do spánkového režimu aby vytvorili virtuálny zhluk uzlov s rovnakými časovými úsekmi načúvania a spania. Uzly, ktoré sú na hranici dvoch zosynchronizovaných zhlukov si môžu vybrať jeden alebo oba časové plány prechodu do spánkového režimu. Na obrázku 2.2 obojsmerná komunikácia znázornená pomocou šípok ilustruje ako koncentrovanie prenosu do kratších aktívnych časových rámcov redukuje nečinné načúvanie.

Na minimalizovanie kolízie uzly používajú mechanizmus štandardu IEEE 802.11 založený na výmene správ RTS a CTS. Keď uzol prehrá v súťaži o médium prechádza do spánku a zobudí sa, keď prijímateľ je voľný a schopný prijímať ďalšie správy. Uzly si nastavujú spánkový interval podľa hodnoty uvedenej v poli duration v každom



Obr. 2.2: Porovnanie pracovných cyklov MAC protokolov [21]

rámci, ktorá indikuje ako dlho bude trvať prenos. Taktiež problematika prepočúvania správ (message overhearing) sa rieši prechodom do spánkového režimu pokiaľ susedné uzly komunikujú. [1][21]

2.5 Timeout MAC (T-MAC)

Na rozdiel od S-MAC na znižovanie nečinného načúvania si uzly dynamicky menia dobu aktívnej komunikácie podľa aktuálneho stavu premávky. Využitím adaptívneho načúvania komunikácie na médiu sa docieli významnej úspory energie a ľahšie sa uzol prispôsobí zaťaženiu v sieti. Uzly využívajúce protokolu T-MAC tvoria virtuálne zhluky, v ktorých si pomocou mechanizmu adaptative timeout (TA) oznamujú zahájenie spiaceho režimu pre zhluk. Na obrázku obr. 2.2 je znázornené ako rovnaká komunikácia u S-MAC protokolu môže byť zrealizovaná počas kratšieho aktívneho režimu pomocou T-MAC. [21]

2.6 Topológia

Výber vhodnej topológie siete závisí na konkrétnej aplikácií. Napríklad na zlepšenie spoľahlivosti celej siete si v projekte Ginseng vybrali stromovú topológiu s oddelenými zhlukmi. V rámci stromového zhluku limitovali počet zariadení na 16, aby znížili premávku pre daný strom, a potom agregované dáta boli s jednotlivých stromov posielané do jednej (v prípade projektu) alebo viacerých základňových staníc

(nazývané sink). Teda výber topológie zohráva veľkú rolu pre splnenie požadovaných kritérií danej siete.

Existujú štyri základné sieťové topológie:

- *Peer-to-Peer* (taktiež nazývané bod bod) – každý uzol komunikuje priamo s ďalšími uzlami bez použitia prostredníka. Preto každé zariadenie dokáže fungovať ako klient a aj ako server.
- *Star* (hviezda) – ide o centralizovanú topológiu, kde uzly - klienti komunikujú s centrálnym uzlom - serverom a nie medzi sebou. Čiže všetka komunikácia ide cez centrálny uzol. Táto topológia je vhodná v prípade malého počtu uzlov.
- *Cluster Tree* (strom) – používa centrálny uzol nazývaný root node (koreňový uzol), ktorý je zriaďovateľom siete a komunikuje s viacerými podradenými uzlami koordinátormi, ktoré spravujú svoju oblasť typu hviezda. Koreňový uzol taktiež inicuje vznik nových podoblastí vymenovaním uzlu do funkcie koordinátora pre danú podoblasť.
- *Mesh* (spleť) – hociktorý uzol môže komunikovať s iným v rámci svojho dosahu. Pomocou skokovej komunikácie sú uzly schopné komunikovať aj s ostatnými uzlami, kedy správa je smerovaná z uzlu na uzol až kým nepríde do cieľa. Vyznačuje sa vlastnosťami samo riadenia a samo liečenia, v prípade poruchy a výpadku uzlov.

2.7 Riadenie topológie

Jednou z techník na zriadenie funkčnej infraštruktúry je riadenie topológie (TC). Po spustení uzlov v sieti si každý uzol prehľadáva svoje okolie, aby zistil s kým môže komunikovať. Riadenie topológie spočíva v možnosti zmeny topológie siete nastavením vysielacieho výkonu uzlov. Motiváciou takéhoto prístupu je zníženie spotrebovanej energie, tak aby uzol nebol obmedzený v komunikácií a boli uspokojené iné vlastnosti komunikácie. V prípade, že sú uzly rozmiestnené pomerne husto je výhodné kontrolovať ich vysielací výkon.

Pri použití lacných rádiových vysielačov v senzorových uzloch môže nastať situácia, že ich dosah nie je možné zmeniť. V takomto prípade sa TC protokoly riadia podľa *homogénneho* prístupu, kedy všetky uzly používajú rovnaký dosah. Pre tento prístup je definovaný CTR (Rozhodujúci dosah prenosu – Critical transmitting range) problém, ktorý spočíva v nájdení najnižšej hodnoty dosahu za predpokladu zabezpečenia konektivity siete. Na druhej strane exituje *nehomogénny* prístup, kedy uzly sú schopné a majú povolené si nastaviť odlišné hodnoty vysielacích dosahov, tak aby neprekročili maximálny povolený dosah. V praxi to znamená nastavenie krátkeho dosahu v oblasti s veľkou hustotou uzlov a viceversa. [1][2]

2.8 Agregácia dát

Hovoriť o agregácií dát má zmysel hlavne u stromovej topológie, pretože obsahuje uzly zodpovedné za preposielanie správ od uzlov patriacich do oblastí, ktoré sú prístupné práve cez tento uzol (parent). Na zredukovanie veľkého množstva správ od senzorov sa získané dáta spracovávajú už v sieti pomocou agregácie na otcovských uzloch. Tieto uzly zbierajú dáta z okolitých senzorových uzlov, ktoré lokálne spracujú a potom pošlú jednu agregovanú správu základňovej stanici. Na konci celého reťazca základňová stanica vypočíta agregovanú hodnotu (priemer) z dát prijatých zo siete. [2]

2.9 Mobilita

Pojem mobilita znamená, že bezdrôtové senzorové uzly sa môžu premiestniť, alebo byť v neustálom pohybe. V takej situácií dochádza k tomu, že senzorový uzol sa dostane mimo dosah svojej pôvodnej oblasti riadenej koordinátorom alebo otcovským uzlom. Keďže chceme, aby bola zaistená konektivita pre tento náš uzol, tak sa musí dostať do ďalšej WSN, ktorá mu umožní pripojenie s rovnakou identifikáciu ako mal v domovskej sieti. Praktickým príkladom môžu byť senzory ako súčasť oblečenia, ktoré slúžia na ochranu pracovníkov pred nebezpečnými látkami rozptýlenými v ovzduší prípadne radiáciou. V rafinérii pohonných hmôt, ktorá sa rozkladá na obrovskej ploche by bolo zbytočné pokryť úplne celú rafinériu, preto by WSN siete fungovali len pre určité oblasti, medzi ktorými sa pracovníci viac či menej pohybujú. Dáta by senzor zaznamenával aj mimo pokrytých oblastí, a ak by senzor zistil dostupnú sieť, tak by nazbierané dáta odoslal na vyhodnotenie. S pomocou mobility sa naskytajú nové možnosti aplikácie a komplexnejšie rozšírenia funkcionalít senzorového uzlu.

3 ŠTANDARDY A ŠPECIFIKÁCIE

Niekoľko štandardov pre bezdrôtové senzorové siete je v súčasnosti schválených a ďalšie sú v štádiu vývoja. Bez existencie štandardov by zariadenia od rôznych výrobcov neboli schopné medzi sebou komunikovať. Použitie štandardov zjednodušuje zavádzanie nových technológií do praxe a má veľký význam pri ich rozširovaní.

3.1 ZigBee

ZigBee špecifikácia poskytuje cenovo výhodné bezdrôtové pripojenie nízko energetických zariadení na krátke vzdialenosti. Špecifikácia umožňuje pripojenie širokej škály zariadení ako v priemysle, tak aj v domácnosti do jedinej kontrolnej siete. V roku 2003 bola predstavená ZigBee špecifikácia od ZigBee Alliance, ktorá je nezisková asociácia združujúca výrobcov elektroniky, súčiastok, vládne regulačné skupiny, ale aj univerzity. ZigBee stavia a doplňuje IEEE 802.15.4 štandard o sieťovú, bezpečnostnú a aplikačnú vrstvu. V rámci špecifikácie bolo vyvinutých niekoľko štandardov [22]:

- Zdravotná starostlivosť (ZigBee Health Care)
- Inteligentné domácnosti (ZigBee Home Automation)
- Automatizácia budov (ZigBee Building Automation)
- Ovládanie spotrebičov(ZigBee Remote Control)
- Monitorovanie a kontrola dodávky energií a vody (ZigBee Smart Energy)
- Vstupné zariadenia ako myši a klávesnice (ZigBee Input Device)
- Monitorovanie dodávky tovaru (ZigBee Retail Services)
- Inovatívne služby pre mobilné zariadenia (ZigBee Telecom Services)
- Univerzálna synchronizácia 3D okuliarov (ZigBee 3D Sync)
- Sieťové zariadenia na rozširovanie ZigBee PRO siete (ZigBee Network Devices)

3.1.1 ZigBee Stack

ZigBee model je zložený z vrstiev podobne ako ISO/OSI model. Každá vrstva vykonáva služby, ktoré poskytuje nadradeným vrstvám a taktiež na plnenie svojej funkcie využíva služby podradených vrstiev. Fyzická vrstva a spojová vrstva je definovaná štandardom IEEE 802.15.4. ZigBee špecifikácia rieši jednotlivé problémy sieťovej vrstvy a ponúka framework pre aplikačnú vrstvu, ktorú tvorí podvrstva APS, ZDO a proprietárne aplikačné objekty. [23]

3.1.2 Adresácia

Na unikátnu globálnu identifikáciu sa podľa IEEE 802.15.4 používajú dlhé 64bitové EUI adresy, ktoré sa skladajú z 24bitového OUI identifikátoru prideleného od IEEE a 40bitovej časti definovanej pri výrobe zariadenia. Takéto adresy sa použijú pri komunikácii medzi jednotlivými senzorovými sieťami. Avšak kvôli úspore sa vrámci lokálnej siete používajú krátke 16bitové adresy, čo dovoľuje skrátiť dĺžku paketu. Komunikácia medzi zariadeniami prebieha pomocou známych adries, ktoré si zariadenie zistí zaslaním žiadosti o adresu vo forme všesmerovej správy na adresu 0xFF.

ZigBee naviac umožňuje použiť nepriame adresovanie pomocou tzv binding a tým zjednodušiť adresovanie správ. PAN koordinátor si vie vytvoriť binding tabuľku, v ktorej má namapované jednotlivé zhluky a koncové zariadenia, ktoré spolu stabilne komunikujú. Každé z týchto spojení predstavuje obojstranný komunikačný kanál. Takéto spojenie vzniká na základe požiadavku uzlu. Pomocou nepriameho adresovanie je možné poslať správu naraz viacerým zariadeniam alebo zhluku bez použitia všesmerovej komunikácie. [23]

3.1.3 Zriaďovanie siete

Prebieha na úrovni sieťovej vrstvy. Zriaďovateľom siete je PAN koordinátor, ktorý sa stará o zisťovanie a údržbu ciest medzi uzlami, riadi pripájanie a odpájanie zariadení do svojej siete a prideľuje adresy novo pripojeným zariadeniam. Krátke 16bitové adresy prideľované sieťovým koordinátorom sa používajú na identifikáciu v lokálnej sieti. Sieťový koordinátor na komunikáciu so zariadeniami na lokálnej úrovni používa krátku adresu 0x00. Nato aby sa zariadenie mohlo pripojiť do siete musí poznať 16bitový sieťový identifikátor tzv PAN ID. Asociácia a disasociácia prebieha podľa IEEE 802.15.4 MAC rozoberanej v druhej kapitole. [24]

ZigBee sieť funguje na rovnakom staticky zvolenom kanále počas celého života siete. Táto vlastnosť robí sieť náchylnou na rušenie a preto nebolo ZigBee odporúčané na použitie v priemyselných aplikáciách. Tento nedostatok odstraňuje nová verzia ZigBee PRO pomocou možnosti preladenia na iný kanál tzv frequency agility. Keď koordinátor zistí zníženú kvalitu spojenia, tak oznámi svojim uzlom zmenu kanálu. Ani toto vylepšenie však nie spoľahlivé v prípade, že používaný kanál sa zaruší natoľko, že uzly neobrdžia správu o zmene kanálu.

3.1.4 Smerovanie

Problematika smerovania úzko závisí na danej topológii siete. K jednoduchej komunikácie medzi uzlom a viacerými uzlami dochádza u topológie typu hviezda. V tomto prípade senzorovým uzlom stačí poznať svojho miestneho koordinátora. Zložitejšie

to je u stromovej topológie, kde sa používa stromové/hierarchické smerovanie. Mesh sieť používa kombináciu stromového smerovania a AODV. AODV používané u ZigBee je lepšie označovať ako Z-AODV, pretože je založené na parametre útlmu cesty (path loss) a nie výbere cesty podľa väčšieho sekvenčného čísla.

Stromové smerovanie – pre určenie cesty nepoužíva smerovaciu tabuľku. Stromové smerovanie je založené na Cskip adresnom algoritme, podľa ktorého hociktoré zariadenie dokáže ľahko určiť, či daná sieťová adresa patrí potomkovi daného zariadenia alebo rodičovi. Takto hociktorý uzol vykonáva jednoduché rozhodovanie, či pošle paket hore alebo dole v rámci stromovej štruktúry. Veľkou výhodou je jednoduchosť algoritmu, malé výpočtové nároky a ušetrenie pamäťovej kapacity absenciou smerovacej tabuľky. V istých prípadoch vie byť toto smerovanie neefektívne a závisí na sformovaní siete. Napríklad dve zariadenia v blízkej vzdialenosti sa pripoja ku vzdialeným koordinátorom tvoriacich rozdielne vetvy siete. Pakety medzi týmito zariadeniami prechádzajú cez mnoho skokov aj keď by medzi nimi bola možná jednoskoková komunikácia.

Spleťové smerovanie – založené na objavovaní ciest a poskytuje väčšiu adaptabilitu. V prípade použitia len stromového smerovania môže nastať situácia, kedy zariadenie nie je schopné posielať pakety alebo zlyhá, čo zapríčiní odrezanie potomkov od zvyšku stromovej štruktúry. Metóda objavovania ciest používa protokol Z-AODV na dosiahnutie smerovania v mesh sieti.

Z výsledkov simulácie v NS2 (sieťový simulátor vhodný na odsimulovanie WSN sietí) podľa [25] stromové smerovanie má rýchlejšiu odozvu pri preposielaní paketov. Potom čo Z-AODV si vytvorí smerovaciu tabuľku, tak počet presmerovaných paketov pre obe metódy je približne rovnaký. Z-AODV si vždy vyberá kratšiu cestu, čo nie je možné pomocou stromového smerovania. Výsledkom simulácie je, že stromové smerovanie je vhodné pre návalové dátové prenosy a Z-AODV pre kontinuálne prenosy. [23][25]

Smerovanie s energetickou bilanciou – predtavuje zahrnutie parametrov spotreby do smerovania. Predchodzie smerovacie metódy nezahŕňajú kontrolu spotreby, preto autori článku [25] navrhujú vylepšenie algoritmov ZigBee smerovania. Pri vyberaní cesty uzol berie do úvahy zostatok energie uzlov danej cesty. Akumulovaný útlm cesty je používaný na výber cesty a vypočíta sa ako:

$$C(P) = \sum_{i=1}^{L=l} C\{[D_i, D_{i+1}]\}, \tag{3.1}$$

kde L je dĺžka cesty P a $C\{[D_i, D_{i+1}]\}$ predstavuje útlm na ceste P z uzlu D_i do D_{i+1}. Útlm danej cesty p_l môžeme vypočítať na základe LQI (Indikácia kvality linky) alebo sa použije 7 ako konštanta. Smerovanie s energetickou bilanciou berie

do úvahy viacero faktorov: stav energie susedných uzlov, vlastný stav energie a kvalitu linky. Na to aby sa susedné uzly dozvedeli rozdelenie energie svojich susedov, uzol v rezervovanom poli rámca posiela hodnotu svojej zostatkovej energie. Z tejto informácie je možné dostať priemernú energiu oblasti E_{avg}. Následne útlm linky p_l medzi uzlom i a susedným j sa vypočíta:

$$p_l = P(\alpha f(E_i, E_j, E_{avg}) + \beta g(LQI)), \tag{3.2}$$

kde $f(E_i, E_j, E_{avg})$ je čiastková funkcia zostatkovej energie a $g(LQI)$ funkcia kvality spojenia. Koeficienty α a β sa nastavujú experimentálne podľa aktuálnej situácie. Výsledkom simulácie je, že smerovanie s energetickou bilanciou je efektívne a predlžuje život uzlov o 20 %. [25]

3.2 WirelessHART

HART (Highway Addressable Remote Transducer) protokol je určený pre posielanie a prijímanie digitálnej informácie cez existujúcu kabeláž v priemysle medzi inteligentnými zariadeniami a monitorovacím alebo kontrolným systémom. WirelessHART stavia na overenom priemyselnom HART štandarde a ponúka bezdrôtové spojenie zachovávajúc si kompatibilitu s existujúcimi HART zariadeniami, príkazmi a nástrojmi. Motiváciou vzniku bolo získavanie dát zo všetkých meracích zariadení, pretože odhadovaných 85 % používaných HART zariadení neposielalo dáta ďalej na vyhodnocovanie často kvôli cene a obtiažnosti pripojenia pomocou káblov. WirelessHART taktiež stavia na fyzickej vrstve štandardu IEEE 802.15.4 (podpora iba pásma 2,4 GHz) a špecifikuje vlastnú sieťovú, transportnú a aplikačnú vrstvu. [26]

3.2.1 Štruktúra WirelessHART siete

Na prenos sa používa TDMA metóda pre prístup k médium, ktorá minimalizuje výskyt kolízie a redukuje spotrebu zariadení. Preferovaná topológia siete je typu spleť, pretože dovoľuje použitie redundantných ciest v prípade rušenia alebo prerušenia spojenia. Všetky sieťové zariadenia musia vedieť smerovať správy v sieti, čiže nie je možné použiť RFD zariadenia ako u ZigBee. Zariadenia tvoriace sieť [27]:

- **Prevádzkové prístroje** pripojené k priemyselným senzorom a aktuátorom využívajúce WirelessHART.
- **Adaptér** umožňuje bezdrôtové pripojenie viacerých stávajúcich HART zariadení.
- **Smerovacie zariadenia** nie sú pripojené k strojom, nemajú senzory ani ovládacie, regulačné výstupy. Používajú sa v prípade potreby vylepšenia bezdrôtového spojenia.

- **Vreckové prístroje** používané na inštaláciu, konfiguráciu, dohľad a údržbu ostatných zariadení.
- **Brány** sprostredkujú komunikáciu medzi prístrojmi a koncovou aplikáciu v zariadení obsluhy pripojenom k podnikovej sieti.
- **Manažér siete** zodpovedný za konfiguráciu siete, časovanie komunikácie, smerovanie správ a monitorovanie siete. Môže byť súčasťou brány alebo koncovej aplikácie. Určuje redundantné cesty podľa oneskorenia, efektivity a spoľahlivosti. Na zachovanie redundantných ciest sa správy striedavo posielajú rôznymi cestami.

3.2.2 Špecifika a rozdiely

WirelessHART používa mechanizmy FHSS a CCA, aby sa zabránilo zníženej kvalite spojenia alebo jeho strate spôsobenej rušením na danom kanále. Princíp FHSS spočíva v rýchlej zmene použitého kanálu v 2,4 GHz pásme počas prenosu. CCA je voliteľnou funkciu, ktorá pred prenosom správy dovoľuje upraviť úroveň vysielaného signálu a umožňuje definovať tzv blacklist kanálov, ktoré sa nemajú používať.

Sieťový manažér si udržuje zoznam všetkých zariadení a stav topológie. Jednotlivé prenosové cesty od zdroja k cieľu zostavuje sieťový manažér na základe informácie získanej z tabuľky susedov každého zariadenia a zároveň sa pýta na aktuálny stav batérie, aby mohol zostaviť optimálnu cestu s ohľadom na šetrenie energie zariadení. Manažér si tak vytvára smerovacie grafy, ktoré obsahujú niekoľko možných spojení medzi dvoma zariadeniami. Požiadavkom je, aby zariadenia mali aspoň dvoch susedov kvôli diverzite ciest a lepšej spoľahlivosti. Zdrojové smerovanie je založené na statickej ceste špecifikovanej odosielateľom použitím zoznamu zariadení kadiaľ má sprava ísť. Toto smerovanie sa používa na testovanie a odstraňovanie chýb jednotlivých ciest.

Špecifikom na transportnej vrstve je možnosť použitia mechanizmu prenosu bloku dát. Medzi zariadením a koncovou aplikáciu sa zostavuje spojovo orientované spojenie s potvrdzovaním správ a ich opakovaním pomocou ARQ, ktoré je užitočné v prípade upozorňovania na významné udalosti. WirelessHART tým spĺňa požiadavku na spoľahlivosť, ktorá ja nevyhnutná pre použite v priemyselnom monitorovaní. Ďalej spoľahlivosť je zabezpečená spomínaným FHSS a diverzitou ciest.

Energetickú efektivitu WirelessHART siete zabezpečuje sieťový manažér tým, že určuje kam a kedy sa majú pakety poslať pomocou grafového smerovania a plánovania spojenia. Dôležité je taktiež zabezpečiť, aby zariadenia nezostali izolované a mali dosah na minimálne dve zariadenia vhodným rozmiestnením. [28]

3.3 6LoWPAN

Základným cieľom je umožniť komunikáciou pomocou IPv6 pre WPAN siete s nízkou spotrebou ako sú napríklad WSN. Za vývojom 6LoWPAN stojí pracovná skupina vrámci IETF. 6LoWPAN umožňuje ľahkú integráciu bezdrôtových vstavaných zariadení s nízkou spotrebou do Internetu a takto ich integráciu do Internetu vecí. Hlavné výhody použitia IP pre tieto zariadenia a aplikácie s nimi spojené sú: použitie existujúcej infraštruktúry IP sietí; interoperabilita medzi LoWPAN a existujúcimi IP zariadeniami pomocou bežných smerovacích techník; otvorené IP štandardy umožňujú ľahšiu inováciu; ľahšie sprístupnenie a ovládanie funkcií LoWPAN pomocou bežných zariadení; použitie moderných zabezpečovacích techník.

Pamäťové a výpočtové obmedzenia WPAN sietí viedli mnohých výrobcov k použitiu proprietárnych protokolov (napríklad ZigBee) vychádzajúc z predpokladu, že IP je príliš pamäťovo a prenosovo náročné. Na umožnenie efektívneho prenosu IPv6 datagramov cez 802.15.4 spojenia slúži 6LoWPAN adaptačná vrstva medzi spojovou a sieťovou IP vrstvou.

3.3.1 Architektúra siete

LoWPAN siete sa pripájajú do IP sietí použitím IP okrajových smerovačov ako tzv stub siete. Okrajové smerovače preposielajú datagramy na sieťovej vrstve, čo je výhodou oproti ostatným ad-hoc sieťovým architektúram ako ZigBee, kde je potrebná komplexná aplikačná brána na prepojenie do ostatných sietí. Aplikačné brány musia rozumieť aplikačným profilom použitým v danej LoWPAN, čiže zmena aplikačného protokolu na senzorovom uzle si vyžiada zmenu na bráne. Z tohto vyplýva veľká výhoda IP smerovačov, ktoré sa nemusia starať o aplikačné protokoly. [29]

3.3.2 6LoWPAN adaptačná vrstva

Rýchlostné, pamäťové obmedzenia LoWPAN a viacskoková povaha komunikácie viedli k vzniku adaptačnej vrstvy. Na podporu IPv6 komunikácia cez LoWPAN sieť je potrebná fragmentácia a kompresia rámcov, pretože dĺžka rámca pre WPAN sieť je limitovaná na 128 bytov oproti 1280 bytom u IPv6. Adaptačná vrstva definuje ako je IPv6 komunikácia prenášaná v 802.15.4 rámcoch a špecifikuje štyri kľúčové elementy [29]:

- *Kompresia hlavičky.* Polia v hlavičke sú stlačené a používajú sa známe hodnoty (pole verzia: 6). Dochádza k vynechaniu polí, ktorých informácia môže byť odvodená z 802.15.4 rámca (napríklad IPv6 adresa).
- *Fragmentácia.* IPv6 pakety sú rozdelené do viacerých rámcov, aby sa dodržala maximálna veľkosť rámca v LoWPAN sieti.

- *Preposielanie na úrovni spojovej vrstvy.* Na preposielanie IPv6 datagramov adaptačná vrstva nesie adresy ďalších rádiových skokov pre jeden IP skok (skok v rámci sieťovej vrstvy).
- *Smerovanie cez spleť 802.15.4 uzlov.* Umožňuje smerovanie medzi PAN sieťami, kde každý rádiový skok sa považuje za IP skok.

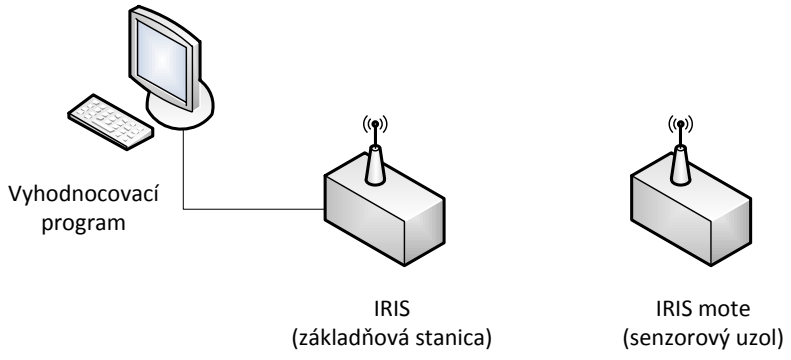
3.3.3 Objavovanie susedov

Technika objavovania susedov (ang. skratka ND) je definovaná pre spojenie medzi susedmi. Používa sa na objavovanie susedov, udržiavanie informácie o dostupnosti, nastavenie defaultných ciest a šírenie konfiguračných parametrov. Posielaním jednosmerných požiadavkou susedom ND vykonáva preklad adries a deteguje nedostupnosť suseda. Smerovač pomocou ND oznamuje uzlom informácie o sieti (prefixy, defaultný limit skokov) a nastavuje defaultné cesty. [29]

4 UPLATŇOVANIE ZÁSAD V OPERAČNÝCH SYSTÉMOCH

Táto kapitola je venovaná vlastnej práci na senzorových uzloch. V základnej konfigurácii sa budeme zaoberať prenosom správ medzi dvoma uzlami a budeme pozorovať vplyv konfigurácie hardvéru na spotrebu uzlu pri použití operačného systému TinyOS. Druhá časť tejto kapitoly rozoberá prístup ďalších dvoch operačných systémov: Contiki a BitCloud. Tieto tri operačné systémy boli v poslednom kroku podrobené hlavnému testu, ktorého výsledky sú zhrnuté v závere práce v kapitole Zhodnotenie výsledkov.

V návrhu sa používajú moduly IRIS XM2110 od Crossbow. Jeden z uzlov pripojení k počítaču funguje ako základňová stanica, ktorá prijíma správy z druhého senzorového uzlu a posiela ich ďalej po sériovej linke do aplikácie, ktorá zobrazuje výsledky obsiahnuté v správe, prípadne programu určenému na záznam správ v textovej podobe do súboru.

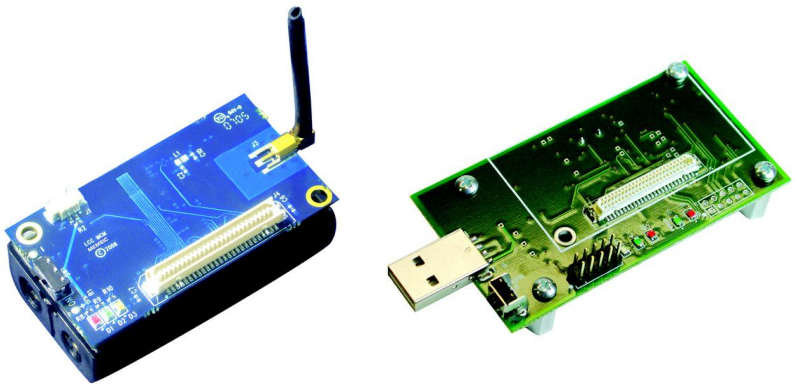


Obr. 4.1: Experimentálna topológia

4.1 Senzorový uzol IRIS

Patrí medzi najnovší modul od spoločnosti Crossbow. Oproti podobným modulom používajúcim IEEE 802.15.4 kompatibilné rádiové čipy prináša trojnásobný rádiový dosah, dvojnásobnú programovú pamäť, nižšiu spotrebu v spiacom režime. Maximálna prenosová rýchlosť je 250 kbps. Procesorová a rádiová platforma od MEMSIC XM2110CB je založená na mikrokontroléri Atmel ATmega 1281 a rádiu AT86RF230. Rozširovací 51 pinový konektor podporuje analógové vstupy, digitálne vstupy a výstupy, I2C, SPI a UART rozhrania. Pomocou tohto konektoru sa najčastejšie pripá-

jajú externé senzorové dosky ako MDA100CB so senzorom teploty, intenzity dopadajúceho svetla a kontaktným poľom, kde je možné pripojiť vlastné senzory prípadne ďalšie vstupy a výstupy. Rozširujúci konektor umožňuje upravenie tohto modulu pre partikulárne aplikácie. V tabuľke 4.1 sú uvedené základné parametre mikrokontroléru a rádia. [30]



Obr. 4.2: IRIS a MIB520 serial/USB rozhranie

Tab. 4.1: Vlastnosti platformy XM2110CB

ATmega1281		**AT86RF230**	
Programová flash	128 kB	Frekvenčné pásmo	2405 – 2480 MHz
Sériová flash	512 kB	Prenosová rýchlosť	250 kbps
RAM	8 kB	Max. výkon	3 dBm
EEPROM	4 kB	Odber prúdu	16 mA (príjem dát)
Č/A prevodník	10 bitový	pri výkone 3 dBm	17 mA (posielanie)
Rozhrania	UART, I2C, SPI	−3 dBm	13 mA
Odber prúdu	8 mA (aktívny režim)	−17 dBm	10 mA
	8 μA (spiaci režim)		

4.2 Testovacia aplikácia

Pre porovnanie operačných systémov je potrebné v každom operačnom systéme implementovať aplikáciu s rovnakou úlohou a nastavenými parametrami. Úlohou tejto aplikácie je získať údaje zo senzoru (hodnota napätia na batériách), následné

poslanie tejto hodnoty do základňovej stanice a prechod do spiaceho režimu, ak to daný operačný systém umožňuje.

Dôležité je spomenúť, že táto testovacia aplikácia neslúži na presné stanovenie spotreby. Metóda na určenie okamžitej spotreby na základe aktuálneho odberu prúdu pre uzol IRIS je popísaná v článku [31]. Efektívnosť a šetrenie energie operačnými systémami je prakticky vyhodnocované na základe poklesu napätia na batériách. Keďže odber prúdu senzorových uzlov je veľmi malý je nutné dlhodobejšie meranie. Ideálne by bolo realizovať meranie aspoň mesiac pre simuláciu skutočnej aplikácie senzorového uzlu. V mojom prípade bola zvolená doba merania dva dni, počas ktorých sa algoritmus merania opakuje každé 2 sekundy, aby jednotlivé výsledné rozdiely poklesu napätia boli markantnejšie, pretože v prípade dlhých spánkových intervalov by za túto dobu merania bolo menej odobranej energie resp. elektrického náboja z celkovej kapacity akumulátoru a tým pádom by sa výsledné hodnoty pre jednotlivé operačné systémy mohli líšiť len málo. V praktickom teste boli použité nabíjateľné batérie typu NiMH (GP 2500)[32], ktoré pred každým testom boli nabité na pôvodnú kapacitu. Takže na krivke vybíjacej charakteristiky sa pohybujeme na začiatku na krátkom úseku, kde napätie poklesne maximálne do 100 mV na jednom článku, kde pokles napätia je strmší, čo taktiež napomáha k realizácii kratšieho merania.

4.3 Operačný systém TinyOS

TinyOS[1] je open source operačný systém vyvíjaný na University of California v Berkeley so spoluprácou s Intel a Crossbow Technology. Operačný systém bol navrhnutý, tak aby spĺňal požiadavky senzorových sietí, v programovacom jazyku nesC (dialekt jazyka C pre vstavené systémy). Bežné aplikácie potrebujú okolo 15 kB pamäte a jadro systému má len 400 B, čiže naozaj ide o miniatúrny systém ako hovorí názov.

V tradičnom pojímaní TinyOS nie je operačný systém, ale skôr aplikačný framework pre vstavané systémy, ktorý výrazne uľahčuje naprogramovanie zariadení na žiadanú funkciu. Pri tvorbe aplikácií sa používa model založený na komponentoch, ktoré sú znovu použiteľné a nezávislé na hardvérovej platforme. [33]

4.3.1 Komponenty

Každá aplikácia pozostáva z grafu poprepájaných komponent, ktoré sú nezávislými výpočtovými entitami. Existujú dva druhy komponent v nesC: moduly a konfigurácie. Moduly poskytujú implementáciu rozhraní. Konfigurácia slúži na zviazanie

[1] použítá verzia TinyOS-2.1.1

ostatných komponent, prepojenie rozhraní použitých komponent s rozhraniami poskynutými inými komponentami. Samotná aplikácia je popísaná komponentou konfigurácie.

Komponenty zahŕňajú špecifickú sadu služieb alebo funkcií, ktoré ponúkajú a používajú prostredníctvom rozhraní. V jednotlivých rozhraniach sú špecifikované príkazy a udalosti. Príkaz je požiadavkou na komponentu aby vykonala službu ako napríklad čítanie zo senzoru. Udalosť slúži na oznámenie vykonania danej služby, napríklad oznámenie o úspechu čítania zo senzoru. Samotná komponenta môže používať alebo poskytovať niekoľko rozhraní a taktiež niekoľko inštancií rovnakého rozhrania. Štruktúra a implementácia komponent bude zrejmejšia z praktických príkladov. Príkazy a udalosti sú vykonané oddelene v angličtine označované ako split phase, teda v dvoch oddelených fázach. Príkazy sa vykonávajú ihneď a program pokračuje na ďalší príkaz kódu. Na druhej strane udalosti oznamujú svoje dokončenie neskôr v čase. [33]

4.3.2 Znižovanie spotreby rádia

Spotrebu spojenú s prenosom dát pomocou rádiového modulu rf230 môžeme ovplyvniť zmenou vysielacieho výkonu ako uvádza tabuľka 4.1

V časových intervaloch, kedy sa neposielajú a neprijímajú správy, rádio načúva na médiu po celý čas s rovnakou spotrebou ako pri prijímaní. Preto je potrebné uspanie rádiového modulu. Hodnota odoberaného prúdu v spiacom režime predstavuje 0,02 μA. [34]

V TinyOS platformovo nezávislá koponenta ActiveMessageC okrem rozhraní na posielanie a prijímanie správ obsahuje aj rozhranie LowPowerListening, ktoré umožňuje nastavenie pracovného cyklu rádiového modulu. Pomocou tohto rozhrania sa nastavuje interval kedy sa má rádio prebudiť a skontrolovať aktivitu na médiu, čiže ak sa práve nepoužíva tak prechádza do úsporného režimu na dobu určenú príkazom `setLocalWakeupInterval(uint16_t intervalMs)`. Príkazom `get` je možné spätne zistiť nastavenú hodnotu intervalu. Rozhranie ponúka aj zistenie a nastavenie Wakeup intervalu vzdialeného uzlu pomocou prenesenej správy, čo slúži na to, aby vzdialený uzol neposielal správu práve vtedy, keď uzol nenačúva na médiu a nie je pripravený prijímať správu. Pre popis rozhrania LowPowerListening viz príloha B.

Spotrebu rádiového modulu ovplyvňuje aj implementácia v aplikácii, teda to ako často sa používa. Preto podľa použitia senzorovej siete je dobré vedieť ako často sa má čítať hodnota zo senzoru. Takže je zrejmé, že spotreba bude nižšia pre dlhšie čítacie intervaly. Taktiež je vhodné posielať viacero nameraných hodnôt ako ich posielať po jednom, teda uplatnenie agregácie dát. Množstvo prenášaných hodnôt je limitované veľkosťou dátovej časti rámca dátovej štruktúry message_t definovanej

v TinyOS. Východzie nastavenie v definícii tejto štruktúry je 28 B, ktoré je možné zmeniť na požadovanú hodnotu nastavením konštanty `TOSH_DATA_LENGTH` v samotnej aplikácii. Pre štandard IEEE 802.15.4 je maximálna veľkosť užitočných dát 118 B pre MAC rámec, pričom sa odporúča použiť 102 B, ponechávajúc tak 25 B na režijné náklady.

Metódu CCA používa rádiový čip na zistenie stavu média, či práve niekto vysiela. V konfiguračnom súbore pre rf230 je možné nastaviť, ktorý mód sa použije. Prvý mód hlási zaneprázdnené médium ak hodnota energie prijatého signálu prekročí nastaviteľnú prahovú hodnotu. Druhý mód funguje na základe detekcie signálu s charakteristikami definovanými vo fyzickej vrstve podľa IEEE 802.15.4, pričom nezáleží na sile signálu. Tretí mód je nastavený ako východzí a využíva kombináciu prvého a druhého módu. Okrem rozoberaných parametrov konfiguračný súbor pre rf230 definuje použitý rádiový kanál (východzí 11 kanál), dobu čakania na potvrdenie doručenia (800 μs) a iné. [33]

4.3.3 Komunikácia na spojovej vrstve

Operačný systém obsahuje implementáciu platformovo nezávislého MAC protokolu IEEE 802.15.4 vo svojej knižnici, ktorý dovoľuje vybrať si medzi beacon a non-beacon režimom. Táto časť TinyOS je vo fáze aktívneho vývoja a väčšina funkcionalít definovaných v štandarde je implementovaná a otestovaná, okrem garantovaných časových slotov a zabezpečovacích služieb. Avšak na použitie tohto MAC protokolu je treba vhodný ovládač rádia a hlavne platformový „zlepujúci kód“, ktorý platforma IRIS neobsahuje. V súčasnosti podporovanými platformami sú TelosB a micaZ používajúce rádiový čip CC2420. Namiesto tohto protokolu sa v TinyOS pre platformu IRIS používa BoX-MAC-2 protokol, ktorý vznikol kombináciou X-MAC a B-MAC. Pred začatím komunikácie vysielacia strana vysiela Wake-up pakety a prijímacia strana kontroluje aktivitu na médiu vždy po uplynutí spiaceho intervalu. Po prijatí Wake-up paketu sa odošle potvrdenie a začína prenos dát.

4.3.4 Úsporné režimy mikrokontroléru

Dnešné mikrokontroléry podporujú viacero napájacích režimov s rôznym odberom prúdu, dobou prebudenia a pripojením periférií. ATmega1281 ponúka 6 režimov [35]: Idle, ADCNRM, Standby, Extended Standby, Power-save, Power-down.

- **Idle** odpája taktovací signál jadra a pamäte flash, zatiaľ čo SPI, UART, I2C, komparátor, Č/A prevodník, časovače a systém prerušenia zostávajú aktívne.
- **ADCNRM** je podobný, ale naviac odpája komunikačné rozhrania UART

a SPI. Tento mód umožňuje merania s väčšou citlivosťou, pretože je zredukované rušenie z prostredia vplývajúce na Č/A prevod.

- **Power-down** vypína externý oscilátor a mikrokontrolér sa prebúdza pomocou externého resetu, pri zhode adresy na I2C rozhraní, externým signálom na pine INT7:4 alebo INT3:0; čiže umožňuje jedine činnosť asynchrónnych častí. Prebudenie je sprevádzané oneskorením (ustálenie oscilátoru).
- **Power-save** je identický s Power-down módom až na to, že ak bol použitý časovač 2, tak tento ostáva bežať počas úsporného režimu a je zdrojom prerušenia spánkového režimu.
- **Standby** je tiež odvodený od Power-down režimu s tým rozdielom, že sa neodpojí externý oscilátor.

Na prechod do jedného z režimov sa nastavujú bity v registre SMCR (Sleep Mode Control Register). Nastavenie týchto bitov je definované v komponente */tos/chips/atm1281/McuSleepC.nc*. Na prechod do úsporných režimov slúži plánovač (scheduler), ktorý v prípade prázdneho frontu úloh vracia mikrokontrolér do úsporného režimu. Komponenta *McuSleepC.nc* poskytuje rozhranie `McuSleep` (viz príloha B), ktoré je zodpovedné za určenie spiaceho režimu bez narušenia chodu podsystémov. Na určenie režimu sa používa algoritmus, ktorý v prípade povoleného prerušenia z jedného z časovačov 0, 1 alebo 3, z rozhrania SPI, aktívneho UART alebo I2C vracia režim Idle. Ak je zapnutý Č/A prevodník tak sa nastaví režim ADCNRM a v prípade nesplnenia žiadnej z predchádzajúcich podmienok sa použije najúspornejší režim Power-down.

V spúšťacej sekvencii TinyOS sa inicializuje plánovač úloh, ktorý je možné inicializovať príkazom `call Scheduler.init()`. Bez inicializácie plánovača by nebolo možné naplánovať spustenie úlohy v najbližšej dobe pomocou príkazu `post` a aktivovanie úsporného režimu. Takto sa v podstate vykonanie úlohy odloží na správnu chvílu.[33]

4.3.5 Aplikácia pre TinyOS

Na demonštráciu bola použitá sada aplikácií Oscilloscope, ktorá je súčasťou TinyOS. Tvoria ju aplikácie napísané v Jave a v nesC. Java Oscilloscope zobrazuje výsledky prijatých hodnôt do grafu prostredníctvom aplikácie Serial Forwarder, ktorá preposiela dáta z nastaveného sériového komunikačného portu.

V module IRIS pracujúcom ako základňová stanica beží aplikácia BaseStation, ktorá hlavne prijíma správy od senzorového uzlu a preposiela ich na sériovú linku, ale taktiež vie poslať správu o zmene vzorkovacej periódy nastaviteľnej v Java Oscilloscope.

Stredobodom pozornosti je softvérové vybavenie senzorového uzlu, v ktorom bola použitá upravená aplikácia Oscilloscope s názvom Osci. Prostredníctvom konfigurácie vlastností hardvéru u tohto uzlu som pozoroval zmeny poklesu napätia na napájacom zdroji. Osci má za úlohu merať úroveň napájacieho napätia v určitých intervaloch a po nameraní definovaného počtu hodnôt odošle správu s týmito hodnotami.

Aplikáciu tvoria tri hlavné súbory s kódom: *OsciC.nc*, *OsciAppC.nc* a *Osci.h*. V hlavičkovom súbore *Osci.h* sa definujú konštanty: interval časovača, počet meraných hodnôt, štruktúra posielanej správy, vysielací výkon rádia podľa [34] a veľkosť dátovej časti posielaného rámca v hlavičkovom súbore *tos/types/message.h*.

OsciAppC.nc je konfiguračná komponenta, kde sa udáva, ktoré komponenty a moduly sa použijú. „App“ v názve naznačuje, že ide o komponentu vrchnej úrovne reprezentujúcu aplikáciu ako celok. Hlavne obsahuje prepojenie rozhraní jednotlivých komponent s rozhraniami komponenty typu modul, v našom prípade *OsciC.nc*.

```
configuration OsciAppC { }
implementation
       {
         components OsciC, MainC, ActiveMessageC, LedsC,
           new AMSenderC(AM_OSCILLOSCOPE),
           new TimerMilliC() as Timer1,
           new TimerMilliC() as Timer2,
           new AMReceiverC(AM_OSCILLOSCOPE);
         components new VoltageC() as VoltageSensor;

         OscilloscopeC.Boot -> MainC;
         OscilloscopeC.RadioControl -> ActiveMessageC;
   ...
       }
```

Ako vidíme v príklade niektoré komponenty sú inicializované pomocou `new`. Jedná sa o generické komponenty alebo inštancie komponenty podobne ako u objektovo orientovaného programovania, čo znamená, že môžeme použiť viac ako jednu inštanciu komponenty.

Modul *OsciC.nc* poskytuje implementáciu samotnej aplikácie, pričom používa rozhrania z konfiguračnej komponenty. Modul sa štandardne delí na časť deklarácie rozhraní a na implementačnú časť. V prípade použitia externých knižníc sa pripájajú pred časťou deklarácie. Štruktúru modulu vidieť v prílohe A. Na tomto príklade vidieť ako nesC používa udalosťami riadený programovací model, pretože aplikácie senzorových uzlov musia reagovať na externé podnety (zmena teploty) alebo udalosti vrámci uzlu. Nadradená komponenta ako je tento modul, keď chce napríklad spustiť

časovač alebo získať hodnotu zo senzoru, tak pomocou `call` volá príkaz podradenej komponenty. Podobne podradené komponenty signalizujú udalosti nadradenej komponente ako napríklad vypršanie časovača. Príkazy a udalosti sú definované v súboroch jednotlivých rozhraní.

4.3.6 Vplyv konfigurácie

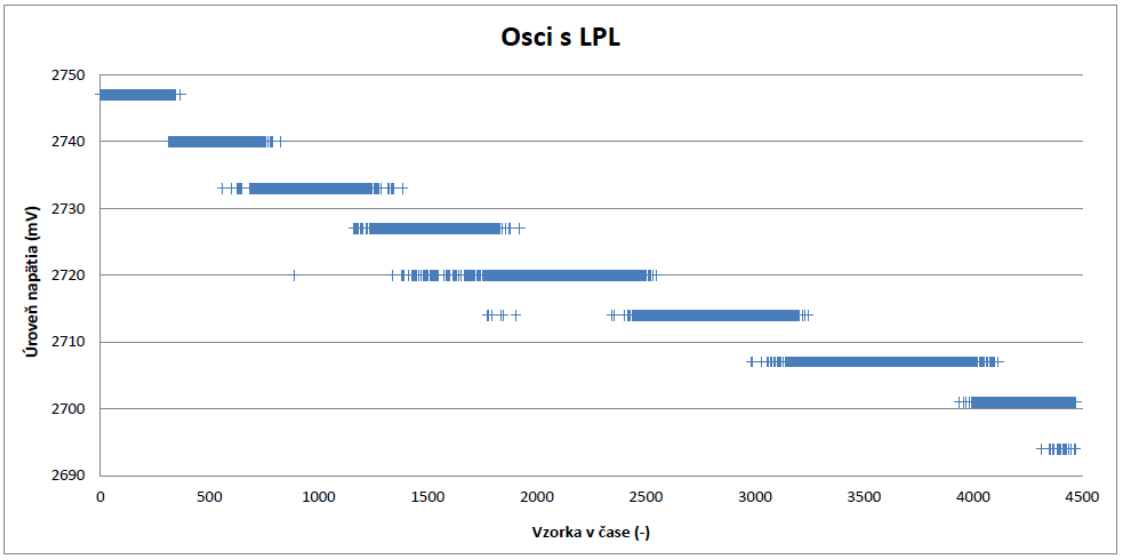
Vplyv konfigurácie hardvéru na spotrebu som pozoroval pomocou monitorovania napájacieho napätia. V tomto prípade nejde o porovnanie presnej spotreby, ale ukázať vplyv konfigurácie na pokles napätia v čase. Na presnejšie určenie spotreby by bolo nutné zasiahnuť do konštrukcie modulu a priamo merať odoberaný prúd z batérie pretekajúci cez vsadený odpor so známou hodnotou odporu. Modul IRIS umožňuje prostredníctvom komponenty *VoltageC.nc* sledovať aktuálne napätie na batériách. Na meranie sa využíva rozhranie čítania pomocou A/Č prevodníku, ktorého konfigurácia je daná modulom *VoltageP.nc*. Hodnota napätia sa odčíta pomocou príkazu `call Voltage.Read()` a výsledok z prevodníku sa spracuje v návratovej udalosti.

```
event void Voltage.readDone(error_t e, uint16_t adc) {
    if (e != SUCCESS)
      {
        adc = 0xffff;
        report_problem();
      }
    local.readings[reading++] =
     (unit16_t)((uint32_t)1100*(uint32_t)1024/(uint32_t)adc);
  }
```

Algoritmus prevodu hodnoty z prevodníku musel byť upravený podľa technickej dokumentácie ATmega1281, pretože odporúčané referenčné napätie v komponente *VoltageC.nc* 1223 mV je pre platformu MICA oproti 1100 mV u IRIS.

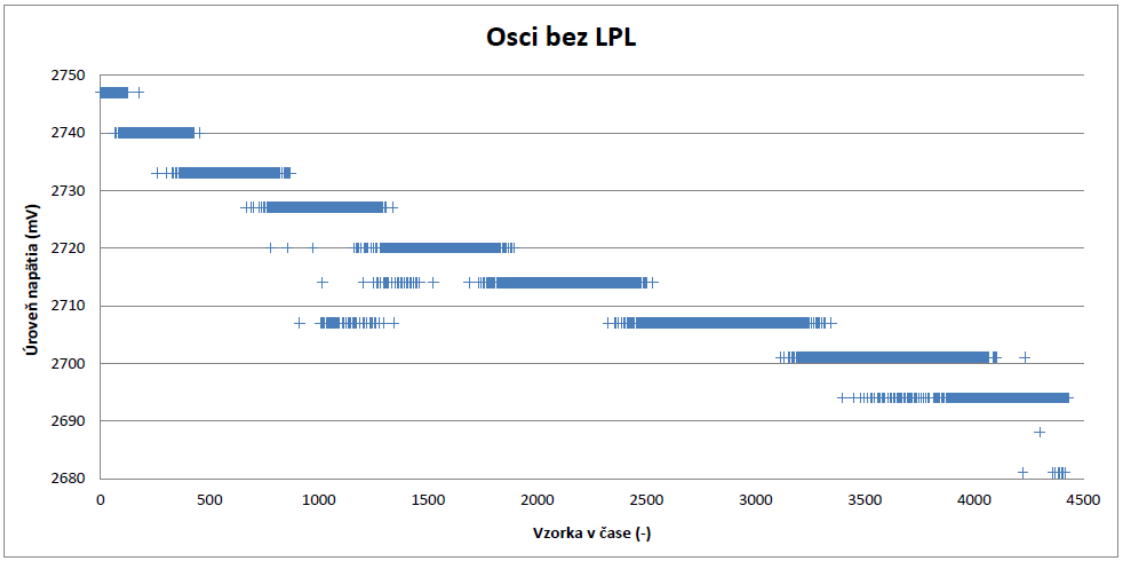
Zrealizované boli dve merania každé po dobu dvoch hodín. Pri každom meraní boli použité nabíjateľné batérie a meranie sa rátalo od rovnakej počiatočnej úrovne napätia.

V prvom meraní bola uplatnená metóda prechodu rádia do spiaceho režimu v intervaloch, kedy nie je potrebné odosielanie dát. Počas tohto intervalu prebieha meranie každých 650 ms. Po nameraní 10 hodnôt sa hodnoty naraz odošlú do základňovej stanice. Posielanie správy a čítanie hodnoty z prevodníku boli naprogramované ako samostatná úloha, čo umožňuje plánovaču realizovať prechod do spánkového režimu mikrokontroléru. Výsledky merania je vidieť na grafe 4.3 a rozdiel napájacieho napätia na začiatku a na konci činí 46 mV.



Obr. 4.3: Namerané hodnoty napätia - uplatnenie úsporných nastavení

V druhom meraní sa použili rovnaké nastavenia pre interval a počet meraní. Neprepínanie rádiového modulu do spiaceho režimu sa docieli nastavením prebúdzacieho intervalu na nulu príkazom `LPL.setLocalWakeupInterval(0)`. Prechody mikrokontroléru do úsporných režimov neboli uskutočnené, pretože použité funkcie neboli napísané ako úlohy.

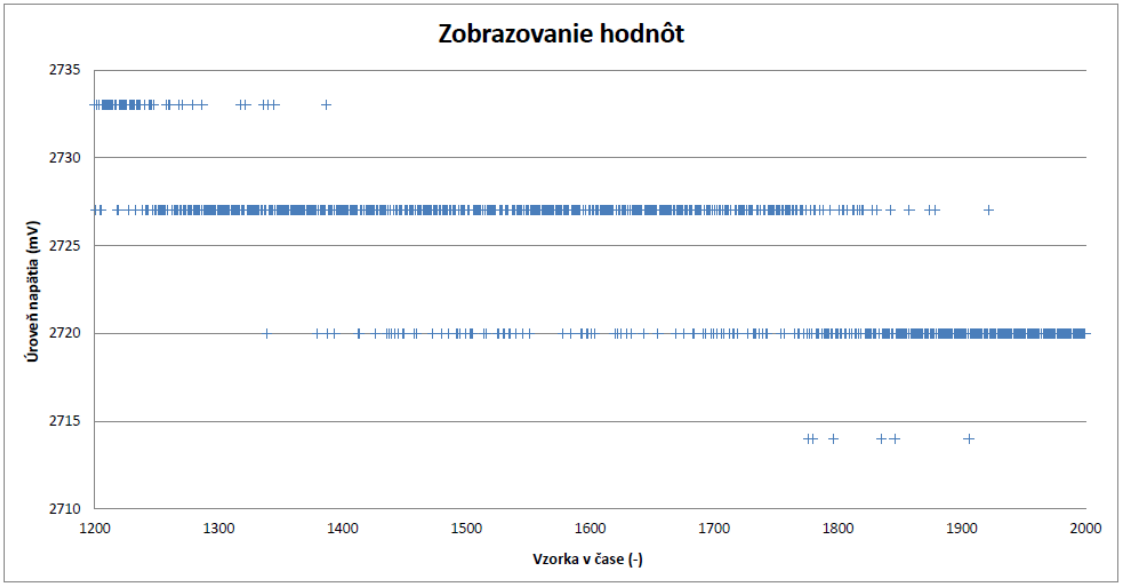


Obr. 4.4: Namerané hodnoty napätia - bez úsporných nastavení

Výsledkom merania bol pokles napájacieho napätia o 53 mV, viz priebeh 4.4. Z týchto výsledkov zároveň aj plynie, že pre porovnanie operačných systémov bude treba dlhodobejšie merania, na ktorých bude vidno jasnejšie rozdiely pri daných

konfiguráciách.

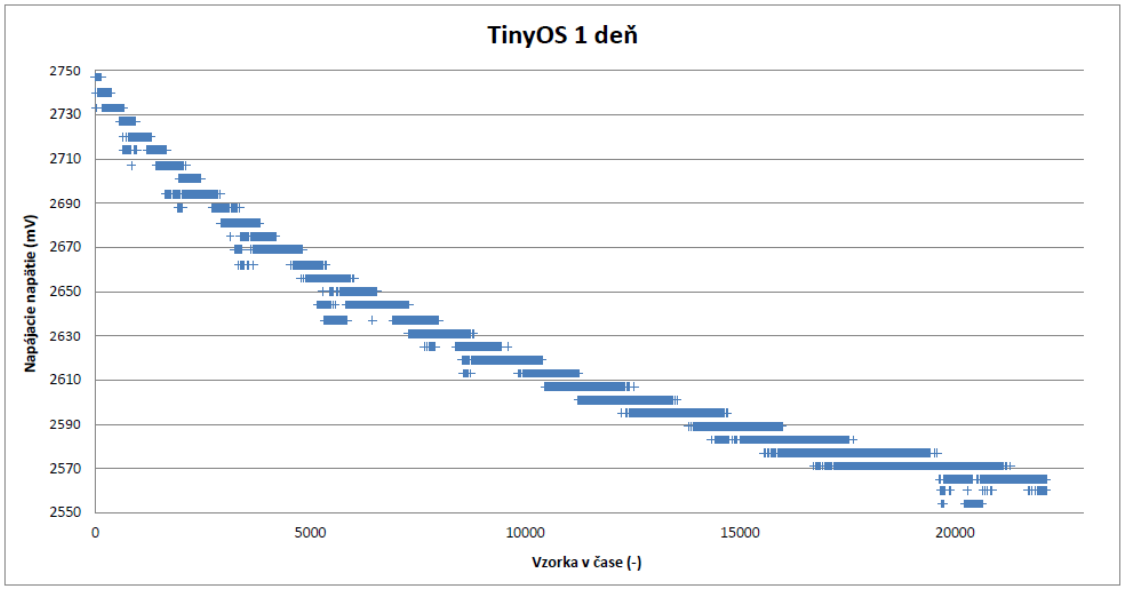
Zobrazenie do grafov bolo realizované vynesením hodnôt z každej správy. Pre správne pochopenie nám poslúži priblížený výrez z nameraných hodnôt 4.5. Najmenší rozdiel hladín napätia je daný použitým výpočtom na prevod výsledku z A/Č prevodníku činiaci 6 mV po zaokrúhlení. To znamená, že keď sa hodnota napätia líši o menej ako 6 mV, tak pri kvantovaní v A/Č prevode dochádza k zaokrúhleniu nahor alebo nadol podľa úrovne získanej z procesu vzorkovania. Napríklad ak máme hodnotu z prevodníku 434, tak dostávame napätie 2595 mV a pre následujúcu hodnotu 435 je prevedené napätie 2589 mV. Pri poklese napätia z jednej úrovne na nižšiu sa hodnota hneď neustáli na nižšej úrovni, ale môže byť striedavo vyhodnotená ako vyššia napäťová úroveň. Preto v zobrazení všetkých hodnôt dochádza k splývaniu v zobrazení, čo však neznamená priradenie jednej hodnote na osi x niekoľkých úrovní napätia. Rovnaký postup bol použitý aj v následujúcich grafoch.



Obr. 4.5: Priblížené zobrazenie hodnôt z grafu

4.3.7 Výsledky testu TinyOS

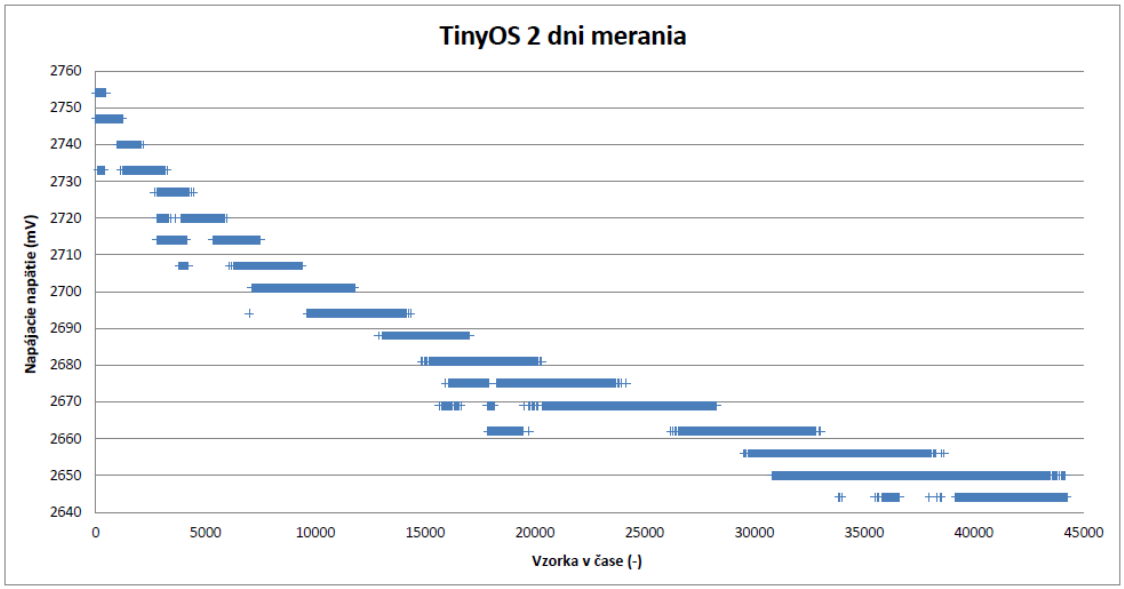
V prvom testovaní bola použitá aplikácia Osci, v ktorej funkcia merania a posielania neboli naprogramované ako úlohy. Pri tomto testovaní som prišiel na to, že ak jednotlivé funkcie nie sú deklarované ako úlohy, tak plánovač úloh si nevytvorí zoznam úloh. Táto chyba mala za následok nevyužitia úsporných režimov a mikrokontrolér stále pracoval v režime Idle. Pokles napätia za jeden deň predstavoval 182 mV. Rýchly pokles a nedržanie sa dlhšiu dobu na jednotlivých napäťových hladinách je vidieť na grafe 4.6.



Obr. 4.6: Namerané výsledky pre TinyOS bez spánkového režimu

Po tomto zistení bola použitá aplikácia LowPowerSensing, ktorá bola vhodne upravená, aby spĺňala požiadavky testovacej aplikácie. Táto aplikácia úspešne využíva spánkového režimu mikrokontroléru a rádia pomocou rozhrania LowPowerListening. Výsledkom merania pre tento operačný systém bol pokles o 110 mV. Pri porovnaní s hodnotou z merania bez spánkového režimu ide o oveľa lepší výsledok. Rozdiel vidieť aj v priebehoch vybíjania 4.6 a 4.7, kedy pri použití spánkového režimu sa napätie drží dlhšiu dobu na rovnakej napäťovej úrovni. V grafe dvojdňového merania sú zobrazené hodnoty z každého druhého paketu, pretože pri spracovaní zobrazovaných výsledkov z aplikácia javaListen[2] pomocou logovacieho skriptu sa zapisoval do súboru každý druhý riadok. Spôsobené to bolo pravdepodobne nedostatočne rýchlym spracovaním a následným zápisom dát skriptom. Toto však nemalo vplyv na meranie a ani na konečný výsledok.

[2]zobrazuje prijaté správy na konzolu



Obr. 4.7: Namerané výsledky pre TinyOS testovaciu aplikáciu

4.4 Contiki OS

Predstavuje open source operačný systém[3] určený pre sieťové vstavané systémy a podľa autorov umožňujúci realizáciu Internetu vecí (Internet of Things). Systém sa vyznačuje svojou portabilitou a je ho možné používať na 8 bitových počítačoch Atari alebo Commodore, dnes používanej architektúre x86, ale hlavne na dnešných mikrokontroléroch používaných práve v senzorových uzloch. Hlavným autorom a vedúcim projektu je Adam Dunkels, ktorý navrhol v roku 2001 uIP najmenší TCP/IP stack. V prípade senzorových sietí to znamená, že môžu byť pripojené do Internetu bez potreby použitia brány realizujúcej preklad medzi komunikačnými protokolmi určenými výhradne pre senzorovú sieť. Contiki poskytuje plne otestovaný IPv6 stack, ktorý spolu s protokolmi určenými pre energeticky efektívne využívanie rádiovej komunikácie ako ContikiMAC, umožňuje batériovo napájaným zariadeniam fungovať ako súčasť IPv6 siete. Operačný systém ako aj užívateľské podprogramy sú napísané v jazyku C.[36]

4.4.1 Štruktúra systému

Systém je rozdelený do dvoch častí: jadro a nahrané programy. Jadro typicky tvorí Contiki kernel, programový zavádzač, podporné knižnice a komunikačný zásobník (stack) s ovládačmi hardvéru. Nad udalosťami riadeným jadrom bežia aplikačné

[3]Contiki OS verzia 2.5

programy, ktoré sú dynamicky zavádzané a uvoľňované za behu systému bez nutnosti prelinkovania. Dynamická štruktúra Contiki predstavuje výhodu oproti TinyOS, v ktorom menšie komponenty sú statické prepojené s jadrom a tak vytvárajú kompletný obraz systému, ktorý po skompilovaní a zavedení nie je možné upraviť.

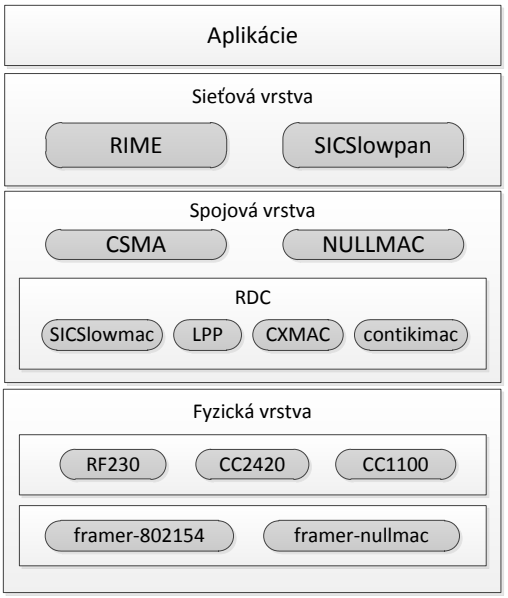
V prípade údržby, modifikácie alebo opravy chýb v existujúcej senzorovej sieti stačí preniesť individuálnu aplikáciu, ktorá je oveľa menšia ako kompletný binárny obraz systému, čo predstavuje úsporu energie potrebnej na prenos do každého jedného uzlu siete. [37]

V tomto systéme sa niekoľko procesov vykonáva súbežne, pričom tieto procesy čakajú na udalosť. Keď nastane udalosť ako vypršanie časovača, prerušenie na vstupno/výstupnom pine alebo vnútorná medziprocesová udalosť, jadro zavolá príslušné obsluhovače udalostí a vykonávajú sa odpovedajúce procesy až do stavu blokovania, kedy čakajú na ďalšiu udalosť. Procesy využívajú koncept protovlákien vyvinutý Adamom Dunkelsom a Oliverom Schmidtom. Protovlákna kombinujú výhody udalosťami riadených systémov a multivláknových systémov. Hlavnými výhodami tohto prístupu je efektívne využitie pamäte (spoločný zásobník pre všetky procesy) a implementovanie mechanizmov riadenia toku vykonávania kódu bez nutnosti použitia komplexných stavových strojov. Vláknové systémy dovoľujú preempciu bežiaceho vlákna - zásobník bežiaceho vlákna sa uloží a vykonávanie iného pred tým prerušeného vlákna môže pokračovať. Na druhú stranu udalosťami riadené systémy a protovlákna používajú jeden zásobník, preto preempcia v Contiki nie je možná. Limitáciou protovlákien oproti udalosťami riadenému modelu je neuloženie automatických lokálnych premenných po blokujúcom čakaní. Lokálne premenné je možné použiť vrámci protovlákien, ale pre zachovanie hodnoty je ich potrebné explicitne uložiť pred vykonaním príkazu čakania na udalosť alebo taktiež sa odporúča použitie globálnych premenných. [38]

4.4.2 Komunikácia

Komunikácia sa v systéme realizuje ako služba, ktorá je nahraditeľná za behu systému, čo umožňuje nahratie viacerých komunikačných zásobníkov a modifikácie zásobníku (stack) na úrovni jednotlivých vrstiev. Prehľad komunikačného zásobníku a používaných protokolov jednotlivých vrstiev je uvedený na obrázku 4.8.

Contiki na rozdiel od ostatných operačných systémov na úrovni sieťovej vrstvy umožňuje výber medzi zásobníkom určeným výhradne pre senzorové siete (RIME) a plne kompatibilným uIPv6 (SICSlowpan) umožňujúcim komunikáciu v celosvetovej sieti Internet.



Obr. 4.8: Prehľad protokolov jednotlivých vrstiev

4.4.3 Šetrenie energie

Contiki neobsahuje žiadne komplexné mechanizmy na šetrenie energie, čiže úspora je závislá na samotných aplikáciách a hlavne na používaných komunikačných protokoloch. Plánovač úloh pomáha aplikáciám pri rozhodovaní, kedy aktivovať úsporný režim poskytnutím informácie o veľkosti radu plánovaných udalostí. [37]

Konfigurácia a používanie rádia

Na úrovni fyzickej vrstvy sa realizuje nastavenie a prístup k prostriedkom rádiového čipu AT86RF230. Ovládač rádiového čipu v operačnom systéme musí obsahovať mapu registrov integrovaného obvodu, aby bolo možné s nimi pracovať (čítať a zapisovať jednotlivé bity). V ovládači sa definujú primitívne funkcie pracujúce práve s týmito registrami, ktoré realizujú napríklad preladenie kanálu, nastavenie vysielacieho výkonu, zistenie intenzity prijímaného signálu (rssi), zistenie stavu prijímača a prechod napríklad do režimu spánku.

V knižnici *rf230bb.h* sú definované minimálne a maximálne nastaviteľné hodnoty vysielacieho výkonu, čísla frekvenčného kanálu, premenné na nastavenie CCA módu. Taktiež obsahuje deklaráciu funkcií pre prácu s rádiom (inicializácia, zistenie a nastavenie kanálu, vysielacieho výkonu, CCA módu), ktoré sú implementované v *rf230bb.c*.

Nastavenie vysielacieho výkonu je možné v registre `PHY_TX_PWR` pomocou bitov označených `TX_PWR`. Maximálny výkon na výstupe je typicky + 3 dBm a je nastaviteľný v rozsahu 20 dB s maximálnou toleranciou ± 3 dB. Odpovedajúci výkon nastavený bitmi `TX_PWR` je uvedený v Table 9-1 datasheetu [34].

Vo východzom nastavení pre IRIS hodnota nastavená na 0, čo znamená maximálny výkon. Avšak výkon je možné meniť dynamicky podľa potreby napríklad pred samotným posielaním predaním atribútu paketového bufru `PACKETBUF_ATTR_RADIO_TXPOWER`, podľa ktorého sa vo funkcii `rf230_transmit()` nastaví požadovaný výkon a po úspešnom poslaní sa nastaví na pôvodnú hodnotu. Ak sa v aplikáci nedefinuje preferovaný komunikačný kanál, tak sa štandardne používa kanál 26. Voľba režimu pre metódu zistenia voľného kanálu (CCA) nie je v ovládači rádia implementovaná a používa sa predvolený režim 1 [4]. Pre zníženie celkovej spotreby rádiového čipu sa používa spiaci režim, kedy dochádza k úplnému vypnutiu a odber prúdu predstavuje len 1 μA [34]. V stave `TRX_OFF`, kedy rádio nie je používané, mikrokontrolér nastavením vstupného pinu `SLP_TR` rádia na vysokú úroveň zaktivuje spiaci režim. Obsah registrov sa v spiacom režime uchováva, zatiaľ čo obsah bufru rámca sa vymaže. Čas prechodu z režimu spánku do `TRX_ON` je v ovládači nastavený na dvojnásobok nominálnej hodnoty 880 μs udávanej výrobcom. Nastavovanie stavov a prechody zabezpečujú funkcie ovládača, teda sa realizujú na úrovni fyzickej vrstvy. Tieto funkcie sú využívané nadradenou vrstvou RDC (Radio Duty Cycling), ktorá spravuje využívanie rádia. Contiki ponúka na výber z viacerých protokolov: ContikiMAC, X-MAC, CX-MAC, LPP a NullRDC. Aj napriek lepšej energetickej efektivite protokolu ContikiMAC sa v prípade IRIS používa CX-MAC, pretože ContikiMAC je špecifikovaný len pre rádio CC2420. CX-MAC je implementovaný na princípoch staršieho X-MAC a funguje na viacerých rádiových čipoch. Na testovanie a pre uzly, ktoré nie sú napájané batériami a stále sú zapnuté je určený NullRDC, ktorý nikdy nevypína rádio. Úlohou RDC je zabezpečiť nízku spotrebu rádia, tým že sa zapína len v prípade prenosu a v periodických kontrolných intervaloch, aby sa skontrolovala aktivita na médiu. Kontrolný interval sa v Contiki nastavuje v Hertzoch (2, 4, 8 a 16 Hz) a pre IRIS je konkrétne `NETSTACK_CONF_RDC_CHANNEL_CHECK_RATE` v *contiki-conf.h* 8 Hz, čo znamená, že osemkrát za sekundu sa kontroluje rádiový kanál alebo inak povedané každých 125 ms sa kontroluje východzí kanál 26. Počas tohto intervalu sa rádio zapne na 6,25 ms.

V praxi podľa zdrojového kódu *cxmac.c* sa pri posielaní paketu použije funkcia `send_packet()`. Na začiatku sa vytvorí hlavička, kde sa zapíše adresa odosielateľa a v prípade ak je zapísaná do paketového bufru adresa príjemcu, tak pakety sa budú

[4] úroveň nosného signálu nad prahovou hodnotou

posielať s adresou príjemcu a nie ako broadcast. Pred samotným posielaním dát sa najprv posiela prúd paketov typu strobe po dobu 128μs, ktorých úlohou je upozorniť príjemcu na dátový prenos. Uzol si uchováva zoznam susedov s časmi posledného stretnutia, kedy od susedného uzla bol prijatý paket typu `STROBE_ACK`. V prípade, že príjemca už je v zozname, tak sa vypočíta čas najbližšieho očakávaného stretnutia a `STROBE` paket sa pošle tesne pred stretnutím. Po odoslaní STROBE paketov prijímač čaká paket `STROBE_ACK` od prijímacej strany, ktorú identifikuje na základe adresy. Ak nebol prijatý `STROBE_ACK`, tak sa odošle ďalší `STROBE`. Pretože druhá strana potrebuje čas na poslanie odpovede, tak po každom odoslaní `STROBE` sa rádio vypína na dobu 128 ns. Po overení, že bol prijatý `STROBE_ACK`, sa pošle paket typu data. Na rozdiel od ContikiMAC protokolu CX-MAC neponúka možnosť súčasného uspania mikrokontroléru s rádiom, kedy pri uspaní rádia sa súčasne volá funkcia `rtimer_arch_sleep(rtimer_clock_t howlong)`, ktorá prepne MCU do úsporného režimu maximálne na 128 ms, čo je dané použitým 8bitovým časovačom.

Využívanie napájacích režimov mikrokontroléru

Contiki nedefinuje špecifické funkcie pre nastavenie napájacích režimov používaného mikrokontroléru, ale tieto sú obsiahnuté v knižniciach pre AVR zariadenia. Spomínaná `rtimer_arch_sleep()` z *rtimer-arch.c* uspí mikrokontroléru spôsobom odpojenia všetkých časovačov okrem jedného, ktorý používa externý oscilátor a slúži na nastavenie doby spánku a následné prebudenie. Volaním `set_sleep_mode()` a `sleep_mode()` sa povolí úsporný režim a MCU prejde do konkrétneho Power-save režimu. Problémov však je, že IRIS nemá defaultne nastavený `RDC_CONF_MCU_SLEEP` prepínač povoľujúci túto funkciu.

4.4.4 Písanie aplikácie (procesu)

Aplikácie v Contiki sa píšu ako procesy, ktoré musia byť v hlavičke programu deklarované. Príkazom `AUTOSTART_PROCESSES()` sa určí proces alebo procesy, ktoré majú byť spustené už pri spustení operačného systému. Výhodou použitia jazyka C je možnosť zahrnúť do programu bežné knižnice ako *stdio.h* pre použitie funkcie `printf()` na výpis hlásení alebo *math.h* pre zložitejšie matematické funkcie. Pre prácu s hardvérom a jeho perifériami sú určené platformovo nezávislé knižnice (napríklad *leds.h* a *battery-sensor.h*), ktoré sú však previazané s hardvérovou abstrakčnou vrstvou (HAL) umožňujúcou priamy prístup k hardvérovým zdrojom. Vrstva HAL používa ovládače, ktoré obsahujú mapu registrov, primitívy zápisu a čítania z registrov a základné funkcie konkrétneho hardvéru. Koncept protovlákien umožňuje daný proces pozastaviť, odloženie vykonania procesu neskôr v čase prípadne blokovanie procesu a čakanie na udalosť. V názornom príklade `PROCESS_WAIT_EVENT()` preruší chod

procesu dovtedy kým nie je poslaná ľubovoľná udalosť do tohto procesu, zatiaľ čo môžu byť vykonávané príkazy iného procesu.
Príkazom `PROCESS_WAIT_EVENT_UNTIL(podmienka)` sa docieli čakania na konkrétnu podmienku ako napríklad vypršanie časovača typu etimer, ktorý po vypršaní posiela procesu udalosť. Contiki ponúka viacej druhov časovačov:

- Timer - sleduje iba svoje vypršanie
- Etimer - posiela udalosť procesu
- Ctimer - volá naprogramovanú funkciu
- Rtimer - meră ubehnutý čas

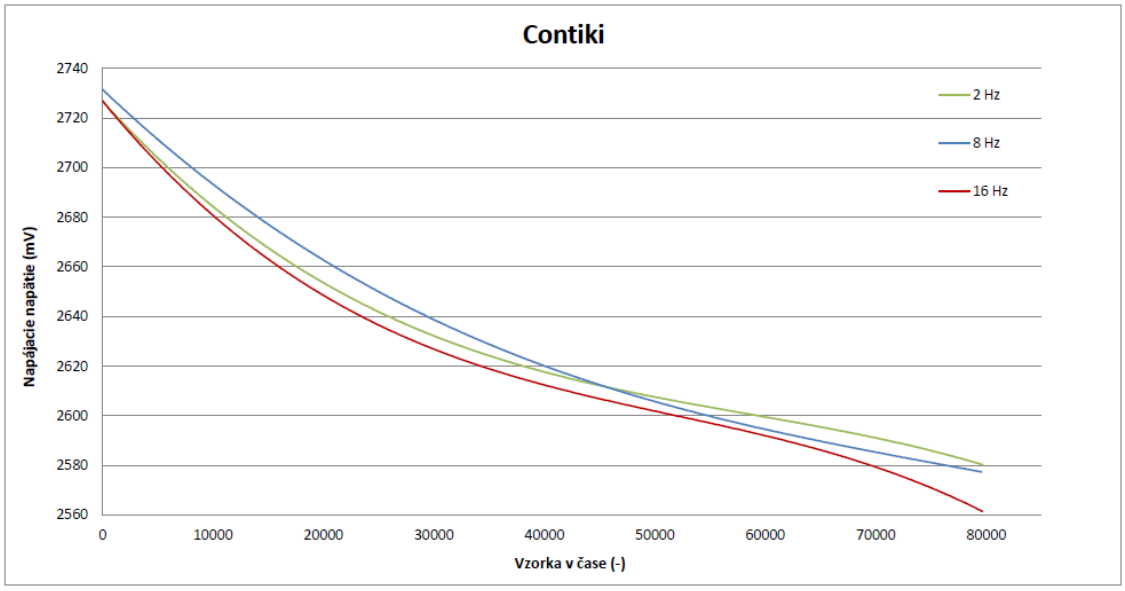
Pred použitím senzorov je potrebného ich aktivovanie, po ktorom je možné vykonávať funkcie definované konkrétnym druhom senzoru. Senzory sa patrične deaktivujú pred ukončením procesu. Aktivácia a deaktivácia senzorov dovoľuje plnú kontrolu nad napájaním senzorov, čiže množstvo ušetrenej energie v prípade senzorov závisí na prístupe aplikácie.

```
#include "contiki.h"
#include <stdio.h>
#include "dev/leds.h"
#include "dev/battery-sensor.h"
/*---------------------------------------------------------------*/
PROCESS(proces_nazov, "Nazov nasho procesu"); //deklarácia procesu
AUTOSTART_PROCESSES(&proces_nazov);
/*---------------------------------------------------------------*/
PROCESS_THREAD(proces_nazov, ev, data) //definovanie nového procesu
{
  PROCESS_EXITHANDLER() // špecifikuje akciu pri ukončení procesu
  PROCESS_BEGIN();
  static struct etimer casovac; // deklarovanie časovača typu udalosť
  leds_on(LEDS_GREEN);
  SENSORS_ACTIVATE(battery_sensor);
  while(1){
     etimer_set(&casovac, 10*CLOCK_SECOND);
     PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&casovac));
     ...
 // čítanie hodnoty AD prevodníku
 // prevod hodnoty alebo poslanie hodnoty z prevodníku
 // všesmerové alebo jednosmerné poslanie správy s hodnotou
 // rozsvietenie diody indikujúce poslanie
  }
  SENSORS_DEACTIVATE(battery_sensor);
  PROCESS_END();}
```

4.4.5 Výsledky testu Contiki

Mechanizmy pre úsporu energie sa uplatňujú v rámci komunikačného protokolu a teda z pohľadu aplikácie nie je možné ovplyvniť mieru úspory. Dostupné je nastavenie periódy kontrolného intervalu rádiového čipu v rozmedzí 62,5 ms – 500 ms. Zrealizované boli tri testy s týmito nastavenými periódami: 500 ms (2 Hz), 125 ms (8 Hz), 62,5 ms (16 Hz). Prakticky to znamená, že pri menšej perióde dochádza k častejšej kontrole rádiového prostredia a tým aj k väčšej spotrebe, čo sa prejavilo aj na výsledných poklesoch: 157 mV, 164 mV, 169 mV.

Neuplatnenie zásady uspávania mikrokontroléru sa prejavilo aj na konečnom výsledku merania, kedy pokles napätia bol 164 mV pre periódu 125 ms, čo v zásade nie je o moc horší výsledok oproti TinyOS. Uspávanie mikrokontroléru je možné doprogramovať pre platformu IRIS vďaka prístupným zdrojovým kódom. Postupný pokles napätia je znázornený v grafe 4.9, ktorého priebeh v zobrazení bez spojníc trendu pre 8 Hz je veľmi podobný 4.6, kde taktiež nebol uspávaný mikrokontrolér. Pre tieto priebehy bez uspávania sa prechádza cez viacej napäťových hladín, na ktorých sa zotrváva kratšiu dobu ako pre priebeh s uspávaním mikrokontroléru 4.7.



Obr. 4.9: Namerané výsledky pre Contiki s troma konfiguráciami

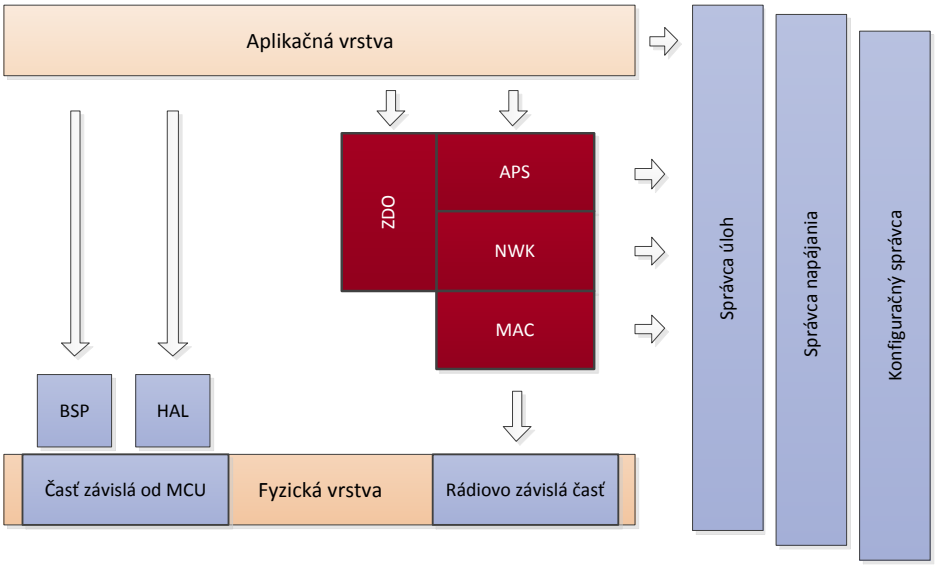
4.5 BitCloud

Predstavuje operačný systém od spoločnosti Atmel určený pre použitie na mikrokontroléroch a rádiových prijímačoch daného výrobcu. Atmel ako člen ZigBee aliancie

implementuje v tomto operačnom systéme ZigBee zásobník, ktorý je plne kompatibilný s ZigBee PRO a ZigBee štandardmi. Keďže celý operačný systém je vyvíjaný v laboratóriách Atmelu, jadro systému a implementácia ZigBee je transparentná pre vývojárov. V počiatkoch vývoja tohto systému nebola prístupná HAL vrstva ani Atmelom pridaná BSP vrstva, čo znamenalo nemožnosť použitia na iných vývojových doskách alebo moduloch ako od Atmelu. Aktuálna verzia už poskytuje zdrojové kódy týchto vrstiev a nie v forme binárnych knižníc. Oficiálne platformy od iných výrobcov nie sú podporované v BitCloud SDK a taktiež nie je možné pridávať podporu iných platform vyvíjaných komunitou, tak ako to je napríklad u Contiki. Preto pre sfunkčnenie BitCloud na platforme IRIS bolo použité upravené SDK[5] od Ing. Ľubomíra Mráza a ďalej bola doplnená BSP a HAL vrstva o potrebné funkcie pre testovaciu aplikáciu (využitie AD prevodníku, prípadné zapnutie/vypnutie periférií).[39]

4.5.1 Štruktúra systému

Vnútorná architektúra systému je rozdelená na logické vrstvy podľa IEEE 802.15.4 a ZigBee, znázornená na obrázku 4.10, na základe ktorého je navrhnutá adresárová štruktúra systému. Podzložky systému predstavujú jednotlivé vrstvy, kde sú umiest-



Obr. 4.10: Architektúra BitCloud podľa [39]

nené zdrojové kódy v prípade HAL, BSP a ConfigServer (predstavujúci Konfiguračného správcu), prípadne konfiguračný súbor a hlavičkové súbory pre všetky ostatné

[5]BitCloud_ZIGBIT_1_11_0

vrstvy. Okrem očakávaných vrstiev BitCloud obsahuje pridané komponenty implementujúce zdieľané služby (správca úloh, správca napájania, konfiguračný správca), ktoré sú dostupné užívateľskej aplikácii a taktiež použiteľné spodnými vrstvami.

V prípade BitCloudu ide o udalosťami riadený systém, v ktorom funkcii náleží asynchrónna notifikácia o výsledku vykonania funkcie realizovaná pomocou návratových volaní. Taktiež rozhrania jednotlivých vrstiev zásobníku (stack) sú definované ako volania funkcie a odpovedajúce návratové volanie. Z toho vyplýva, že každá vrstva definuje návratové funkcie pre nižšie vrstvy a sama vyvoláva návratové funkcie definované nadradenými vrstvami. Príkladom užívateľom definovanej návratovej funkcie je obsluhovač úloh zodpovedný za vykonávanie kódu aplikačnej vrstvy. Typicky sa obsluhovač skladá z prepínača medzi definovanými stavmi.

4.5.2 Konfiguračný správca

Úlohou konfiguračného správcu je nastaviť východzie hodnoty registrov rádia a mikrokontroléru pri štarte systému. Všetky potrebné hodnoty ako veľkosť bufru rámca, tabuľky susedov, maximálny počet skokov paketu a vysielací výkon s ohľadom na jednotlivé vrstvy systému sú definované v hlavičkovom súbore *csDefaults.h*. Hodnoty jednotlivých parametrov závislých od požiadavkou konkrétnej aplikácie sa modifikujú v hlavnom konfiguračnom súbore aplikácie *configuration.h*.

Obraz operačného systému sa väčšinou kompiluje spolu s časťami kódu určeného výhradne pre jednotlivé typy zariadení v sieti (koordinátor - ZC, smerovač - ZR, koncové zariadenie - ED), čo umožňuje zmenu úloh zariadení v rozostavenej živej sieti v prípade zlyhania napríklad smerovaču. Úloha zariadenia sa nastaví v konfiguračnom súbore aplikácie spolu so 64 bitovým unikátnym identifikátorom zariadenia `CS_UID` a sieťovou adresou, ktorá sa pre koordinátora typicky volí nulová. Taktiež je potrebné zvoliť identifikátor siete PAN, ktorú koordinátor vytvorí a do ktorej sa koncové zariadenie má pripojiť.

Dôležitými konfiguračnými parametrami pre rádio sú komunikačný kanál a výkonová úroveň. Kanál sa volí na základe `CS_CHANNEL_MASK`, pomocou ktorej je možné špecifikovať rozsah kanálov, ktoré sa pred pripojením oskenujú a v prípade zhody identifikátoru PAN uzol začne komunikovať s koordinátorom na danom kanále.

4.5.3 Šetrenie energie

Správca napájania využíva definované rutiny zodpovedné za vypnutie všetkých komponent zásobníku a zachovanie stavu systému pred stavom spánku a následné obnovenie stavu systému po prebudení hardvéru. Čiže úspora energie z pohľadu operačného systému a aplikácie sa realizuje prepnutím do stavu spánku, ktorý volá funkcie

uspania zariadenia. Z tohto vyplýva, že závisí od danej aplikácie, kedy sa prepne do stavu spánku volaním `ZDO_SleepReq()` s parametrami ako je doba spánku. Tento parameter sa typicky volí 10 s v konfiguračnom súbore aplikácie. Táto hodnota nie je smerodajná len pre koncové zariadenie, ale pracuje s ňou aj koordinátor, ktorý potom vie kedy môže očakávať správy od koncových zariadení a hlavne vie kedy dané zariadenie nespí, aby mu tiež mohol poslať správu.

Samotné nastavenie spánkového režimu vykonáva `halPowerOff()` v *halSleep.c*, kde v prípade použitia interného oscilátoru sa aktivuje power-save režim ak už bol spustený spánkový časovač alebo power-down režim ak ešte nebol spustený časovač. V prípade použitia externého oscilátoru sa používa jedine power-save režim, počas ktorého môže bežať Timer2/Counter. Mikrokontrolér pri nastavovaní spánkového režimu tiež dáva povel rádiu na uspanie nastavením logickej jednotky na pine PORTB 7, ktorý je pripojený na SLP_TR rádia.

Zapínanie a vypínanie periférií sa realizuje volaním funkcie `bspOnPeriphery(id)` resp. `bspOffPeriphery(id)` z BSP vrstvy s identifikátorom senzoru alebo inej periférie. Použitie a naprogramovanie tejto funkcie sa odvíja od zapojenia senzoru do konkrétneho I/O portu, ktorý sa nastaví ako výstupný a na logickú jednotku (napríklad `GPIO_7_set()`) pred použitím a vypne sa po realizácii napríklad AČ prevodu. Sfunkčnenie periférií ako led diody, LCD, gombíkov a senzorov sa v testovacej aplikácii *Lowpower*, popisovanej ďalej, vykonáva v inicializačnej časti programu volaním `boardAbstracionOpen()` a vypnutie po úspešnom prenose dát v stave `PERIPHERY_CLOSING` volaním `boardAbstracionClose()`.

4.5.4 Písanie aplikácie v BitCloud

Aplikácie v tomto operačnom systéme sa skladajú z hlavného zdrojového kódu (*lowpower.c*) a podčastí pre konkrétny typ zariadenia (*coodinator.c* a *enddevice.c*). Celá aplikácia je v porovnaní s predošlými operačnými systémami komplexnejšia a tvorí ju viacej riadkov kódu, preto má zmysel popísať len jadro a hlavné časti kódu z aplikačnej vrstvy.

Každá aplikácia musí obsahovať funkciu main, ktorá inicializuje mikrokontrolér a následne v nekonečnom cykle `SYS_RunTask()` predáva riadenie chodu plánovaču úloh, ktorý volá postupne obsluhovače vrstiev OS začínajúc aplikačným obsluhovačom `APL_TaskHandler()` v *lowpower.c* uvedenom v príklade. Tento obsluhovač má za úlohu riadenie prechodu do jednotlivých stavov programu na základe zdieľanej globálnej premennej `appState`. V rámci stavov sa vykonávajú príslušné úlohy, ktoré zároveň menia hodnotu `appState`, aby pri volaní obsluhovača sa prešlo do nasledujúceho stavu.

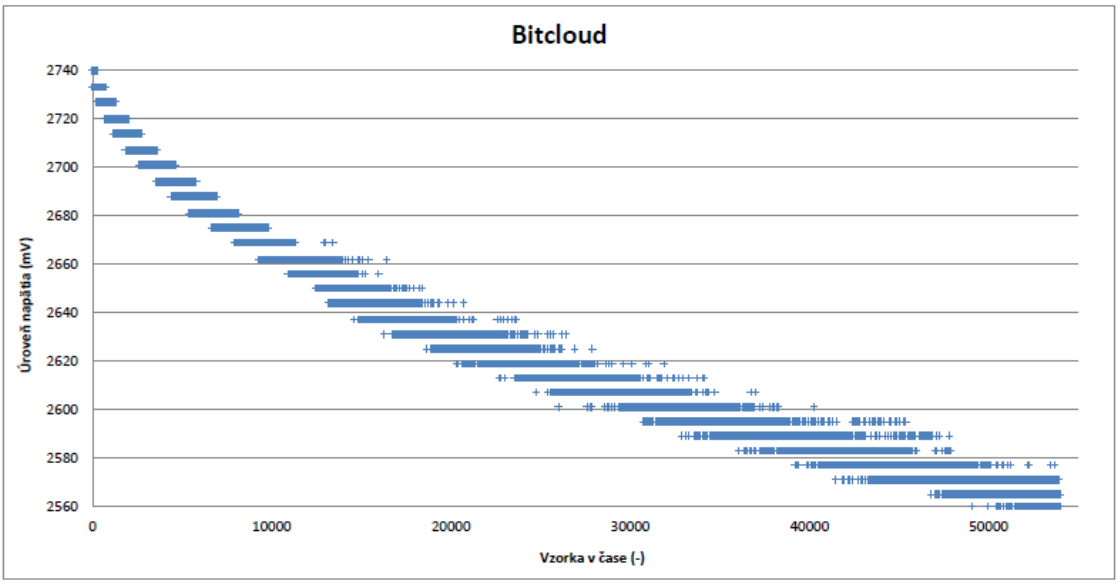
```
int main(void)
{
  SYS_SysInit();
  for(;;) { SYS_RunTask();}
}

void APL_TaskHandler(void)
{
 switch (appState)
 {
  case APP_STATE_INITIAL: // Zapnutie periférií, zapísanie adries
   initApp(); // nastavenie spoločných a špecifických
   break; // parametrov podľa typu zariadenia

  case APP_STATE_NETWORK_JOINING: // Coordinator / EndDevice
   startNetwork(); // Zriadenia / pripojenie do siete
   break;

  case APP_STATE_NETWORK_JOINED: // Po pripojení sa predá
   if (ops && ops->taskHandler) // kontrola obsluhovača
        ops->taskHandler(); // podľa typu zariadenia
   break;
  }
}
```

Po úspešnom pripojení ED do siete vytvorenej ZC sa riadenie chodu programu predáva obsluhovaču implementovanom v `enddevice.c`. Štruktúra obsluhovača je uvedená v prílohe C.

4.5.5 Výsledky testu BitCloud

V prípade BitCloud bola použitá aplikácia Lowpower, ktorá používa spomínané mechanizmy úspory. Z prístupu k úspore systému, by sa dalo čakať, že dopadne najlepšie v praktickom teste, keďže dochádza k uspaniu celého zariadenia po vykonaní úloh na nastavenú dobu dvoch sekúnd. Konkrétne dochádza k odpojeniu rádia, periférií a prechodu mikrokontroléru do power-save režimu.

Prvé dva testy boli realizované so zapnutým potvrdzovaním na aplikačnej vrstve, kedy podľa meraní z [31] IRIS odoberá 0,5 s prúd 12,5 mA a 0,6 s prúd 10,3 mA zatiaľ čo po úspešnom odoslaní správy sa hneď senzorový uzol neuspí, ale ešte čaká na potvrdenie od základňovej stanice. Tento druh potvrdzovania vedie k preneseniu

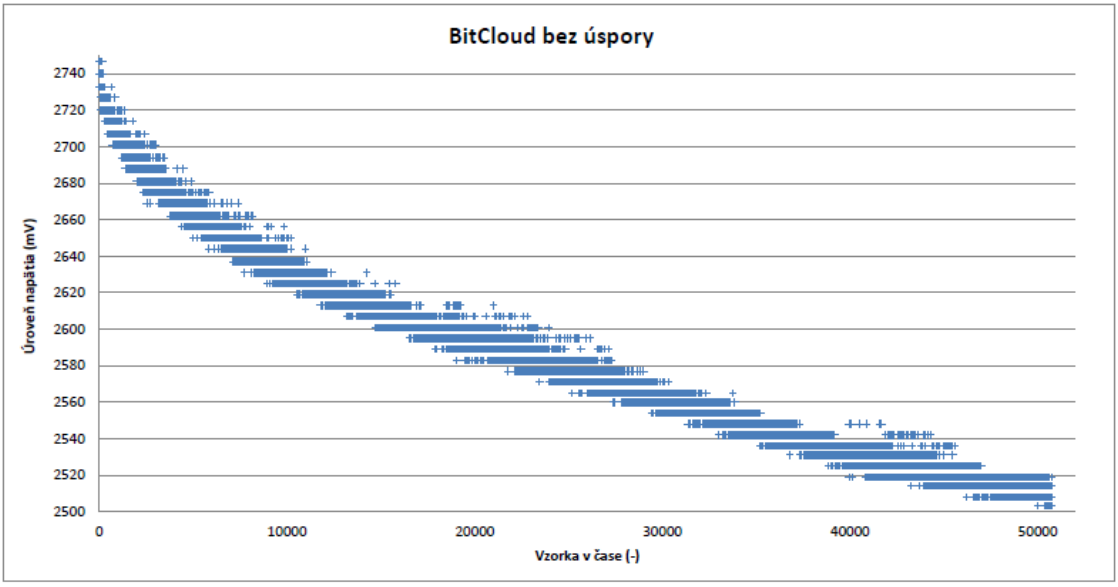


Obr. 4.11: Výsledky testovacej aplikácie Lowpower

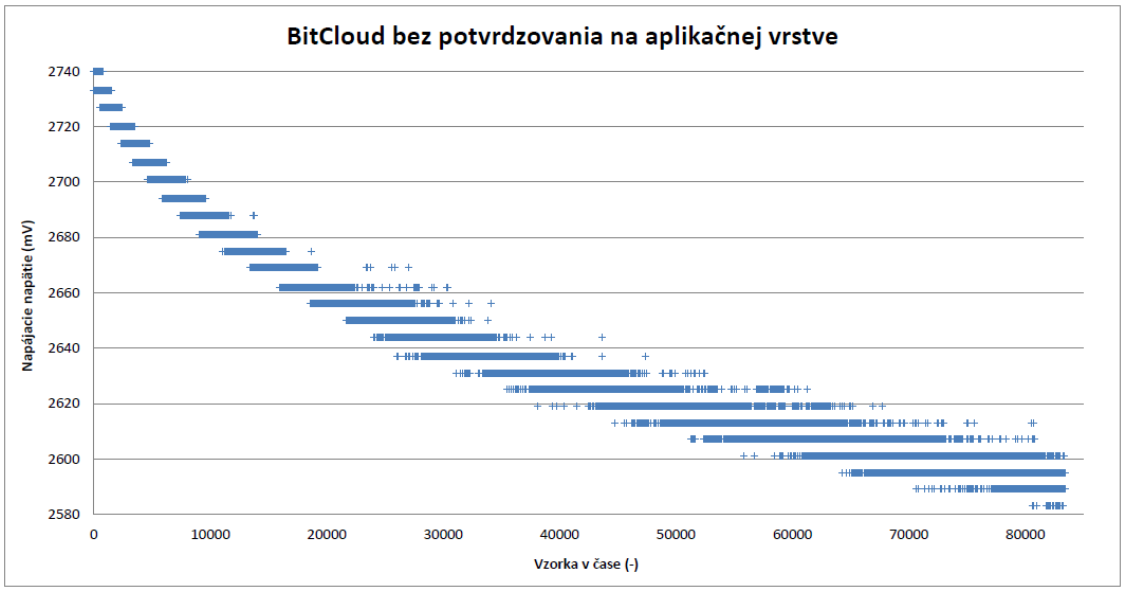
menšieho počtu správ za rovnakú dobu a taktiež k väčšej spotrebe, pretože uzol zostáva aktívny dlhšiu dobu. V tomto prípade je časový rozdiel medzi dvoma prijatými správami 3,2 s.

V prvom teste pokleslo napätie o 173 mV, čo je podstatne horší výsledok oproti TinyOS, aj keď dochádza k uspaniu celého zariadenia. Zmerané hodnoty sú zobrazené v grafe 4.11 Dôvodov horšieho výsledku je hneď niekoľko. V prípade BitCloudu ide o komplexnejší systém, ktorý striktne dodržiava vrstvovú architektúru. Prakticky to znamená, že napríklad pri získaní hodnoty zo senzoru sa postupne musí prejsť cez jednotlivé vrstvy až kým sa hodnota viacerými návratovými volaniami nedostane až do aplikačnej vrstvy. Použitý ZigBee štandard striktne vyžaduje potvrdzovanie na spojovej vrstve a hlavným dôvodom je použitie aplikačného potvrdzovania.

V druhom teste aplikácia Lowpower bola upravená tak, aby nedochádzalo k spánkovému režimu a bolo vidieť rozdiel oproti uplatneniu úsporného prístupu 4.12. V tomto prípade zostáva mikrokontrolér a rádio stále zapnuté. Meranie a posielanie hodnoty napätia batérií sa deje po vypršaní časovača, ktorého interval bol nastavený na 3200 ms. Tento interval bol zvolený, tak aby čas medzi prijatými správami bol rovnaký ako pri použití spánkového režimu, kedy interval medzi jednotlivými správami je navýšený čakaním na potvrdenie a samotným prechodom do a z spánkového režimu. V tomto teste bol pokles napätia 225 mV, na charakteristike vybíjania sa pomaly dostávame už do časti, kde pokles nie je taký prudký, preto táto hodnota je len o 52 mV väčšia oproti predchodziemu testu.



Obr. 4.12: Výsledky upravenej Lowpower



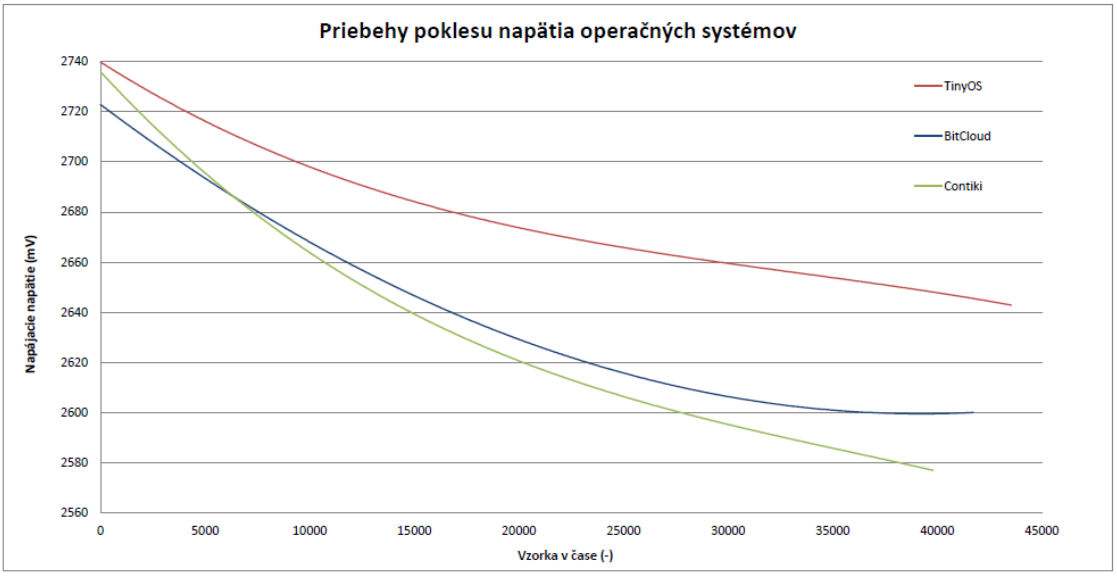
Obr. 4.13: Výsledky Lowpower bez potvrdzovania

V treťom teste bolo vypnuté spomínané aplikačné potvrdzovanie, ktoré sa u ostatných operačných systémov nepoužíva. Týmto sa dosiahne rozdielu medzi správami dvoch sekúnd a prenesenie väčšieho počtu správ za dobu dvoch dní. Z priebehu 4.13 vidíme, že aj keď sa prenieslo viac správ ako v predošlých dvoch meraniach, tak pokles napätia bol nižší a to 144 mV.

4.6 Zhodnotenie výsledkov

Spoločné porovnanie operačných systémov vychádza z výsledkov testovacej aplikácie pre všetky operačné systémy s uplatňovaním úsporných režimov.

Pred samotným zhodnotením meraní som očakával najlepšie výsledky u operačných systémov TinyOS a BitCloud. TinyOS realizuje uspávanie zariadenia nezávislými technikami pre rádiovú časť a pre mikrokontrolér použitého modulu IRIS, tým pádom je možné lepšie nastaviť parametre pre uspávanie na základe požiadavkou aplikácie. U operačného systému BitCloud dochádza k uspaniu celého zariadenia jednou technikou. Vždy po vykonaní naprogramovaných úloh, prechádza do stavu spánku na dobu nastavenú v konfiguračnom súbore. Tento prístup garantuje významnú úsporu a pravidelné prebúdzanie zariadenia podľa zvoleného intervalu, čo slúži zároveň k časovej synchronizácii medzi uzlami v sieti. Z porovnávaných operačných systémov sa očakáva najhorší výsledok od Contiki, vzhľadom na to, že pre platformu IRIS používa len uspávanie rádia na základe parametru popisovaného v kapitole venovanej tomuto operačnému systému, čiže nie je možné prispôsobiť prechod do úsporného režimu podľa aplikácie a tým zabezpečiť dlhšie spánkové intervaly.



Obr. 4.14: Porovnanie výsledkov skúmaných operačných systémov

V konečnom zhodnotení systémov v prípade Contiki boli použité výsledky z testu s nastavenou kontrolnou periódou rádiového prostredia 125 ms (8 Hz), ktorá sa používa ako východzia hodnota. V prípade BitCloud som použil výsledky z merania bez potvrdzovania na aplikačnej vrstve, tak aby mohol byť tento systém porovnaný s ostatnými. Získané hodnoty v meraniach pre Contiki a BitCloud boli zredukované

na polovicu vynechaním hodnôt s nepárnym indexom, tak ako to bolo spôsobené u TinyOS pri zázname merania.

Graf 4.14 zobrazuje spojnice trendu z nameraných hodnôt. Z týchto kriviek vidíme ako postupne prebiehalo vybíjanie batérií v IRIS za použitia rozoberaných operačných systémov. Z výsledkov vychádza TinyOS ako najmenej energeticky náročný a Contiki na druhú stranu ako najnáročnejší systém a to kvôli spomínaným očakávaným dôvodom. Číselne vyjadrené výsledky poklesu oproti počiatočnej hodnote napätia napájania je v tabuľke 4.2

Tab. 4.2: Vlastnosti operačných systémov

Operačný systém	Výsledný pokles napätia (mV)	Priemerný čas medzi správami(s)	Pamäť flash (Bytov)	
			Stanica	Uzol
TinyOS	110	2,005	16554	22186
Contiki	164	2,169	26530	26610
BitCloud	151	2,071	18759	18759

V každom systéme bol nastavený časovač pre vykonanie algoritmu zmerania a poslania na dve sekundy. Tento čas môže byť však predĺžený samotným prenosom (napr. výskyt bitových chýb), spracovaním a zápisom do súboru na prijímacej strane a nie je vždy konštantný. V prípade Contiki je časový rozdiel medzi po sebe prijatými správami 2 sekundy a vo výnimočnom prípade 3 sekundy. Pre BitCloud je tento časový rozdiel 2 sekundy s menšími odchýlkami medzi správami a najpresnejšie dodržaný interval medzi správami dvoch sekúnd má TinyOS. Presné priemerné hodnoty sú uvedené v tabuľke 4.2. Tieto časové rozdiely majú za dôsledok rozdielneho počtu zmeraných hodnôt za danú dobu 2 dní pre jednotlivé operačné systémy. Ďalej v tabuľke ešte uvádzam prehľad veľkostí aplikácií pre operačné systémy rozdelené podľa role uzlov.

5 ZÁVER

Práca rozoberá pomerne aktuálnu problematiku bezdrôtových senzorových sietí, ktoré predstavujú technológiu budúcnosti. Úvodná časť práce je venovaná charakteristike tohto typu sietí a ich využitiu. V práci je rozoberaný štandard IEEE 802.15.4, ktorý je východiskovým bodom pre vývoj a vznik ďalších štandardov riešiacich problematiku vrstiev nadradených spojovej vrstve. Medzi tieto štandardy patrí ZigBee, WirelessHART a 6LoWPAN, ktoré sú popísané v tretej kapitole.

Cieľom tejto práce bolo preskúmať zásady konfigurácie bezdrôtových senzorových sietí, ktoré majú hlavný vplyv na energetickú náročnosť senzorových uzlov a efektívne využívanie hardvérových prostriedkov. Uplatňovanie týchto zásad a ich efektívnosť som zisťoval u troch aktuálne najpoužívanejších operačných systémov: TinyOS, Contiki a BitCloud. Pre porovnanie efektívnosti šetrenia energie batérií bola vybraná metóda založená na miere vybitia batérií oproti východziemu stavu, ktorú indikuje aktuálne dodávané napätie. Na začiatku praktického testovania bolo potrebné jednotlivé operačné systémy rozbehnúť na platforme IRIS a nájsť správne parametre a intervaly posielania správ pre testovaciu aplikáciu. Tieto parametre boli zvolené, tak aby testovacia aplikácia nemusela bežať viac ako dva dni pre dosiahnutie zrovnateľných výsledkov a tým pádom mohlo byť zrealizovaných viacej meraní. Všeobecný návrh aplikácie je popísaný na začiatku štvrtej kapitoly. Operačné systémy podporujú platformu IRIS na rôznej úrovni a nie všetky funkcie ponúkané systémom sú implementované.

V TinyOS na porovnanie vplyvu konfigurácie som upravil modelovú aplikáciu Oscilloscope a na medzi systémové testovanie som vychádzal z aplikácie LowPowerSensing, ktorej základný algoritmus musel byť doplnený, tak aby vyhovoval všeobecnému návrhu testovacej aplikácie. Dôvodom použitia LowPowerSensing bolo vhodnejšie uplatňovanie úsporných mechanizmov a nepoužitie Java aplikácie na vykresľovanie zmeraných hodnôt. Prijímané správy základňovou stanicou boli spracované napísaným skriptom, ktorý realizuje separáciu a konverziu dát z prijímanej správy do čitateľnej formy a zapisuje správy označené časovou známkou do súboru. Výsledný pokles oproti počiatočnej hodnote predstavoval 4 %.

Pri vytváraní Contiki testovacej aplikácie som vychádzal z jednoduchej ukážky posielania správ využívajúcej RIME zásobník. Na spracovanie príchodzích správ v tomto operačnom systéme nie je potrebné použitie aplikácie preposielajúcej správy zo sériového rozhrania na konzolu ako to bolo potrebné v TinyOS. Príchodzie správy sa priamo vypisujú do konzoly, z ktorej sú pomocou programu serialdump zaznamenávané do logovacieho súboru. Výsledný pokles oproti počiatočnej hodnote predstavoval 5,97 %.

BitCloud v oficiálnom SDK podporuje platformy len od Atmelu. Z tohoto dôvodu

som nadviazal na upravené SDK od pána Ing. Mráza podporujúce IRIS. Do tohto SDK pre potreby testovacej aplikácie som doprogramoval ovládač A/Č prevodníku a možnosť využitia senzoru batérie v aplikačnej vrstve. Pre testovacie účely som upravil vzorovú aplikáciu Lowpower, tak aby vedela použiť senzor batérie u IRIS a hodnota sa správne predala až do aplikačnej vrstvy. Zmerané hodnoty boli zaznamenávané pomocou programu Terminal, aby mohli byť ďalej spracované a vyhodnotené s výsledným poklesom 5,51 %.

Z jednotlivých výsledkov meraní som prišiel k záveru, že na prvý pohľad komplikovaný operačný systém TinyOS založený na štruktúre previazaných komponent a používajúci jazyk nesC dokáže byť oveľa efektívnejší ako BitCloud, ktorý na aplikačnej vrstve využíva stavový automat a je napísaný v jazyku C, ktorý je oveľa prívetivejší pre vývojárov. BitCloud aj napriek kompletnému uspaniu hardvérovej platformy, ktoré je riadené z aplikačnej vrstvy, je energeticky náročnejší ako TinyOS. Contiki pre platformu IRIS aj napriek nevyužitiu uspania mikrokontroléru, dosiahol len o 7,9 % horšieho výsledku v porovnaní s BitCloud. Contiki je najmladším z týchto systémov a predstavuje reálnu konkurenciu TinyOS, pretože nachádza veľkú obľubu medzi vývojármi, ktorý systém neustále vylepšujú a záleží od komunity kam ďalej sa bude uberať jeho vývoj. Na druhej strane vývoj BitCloudu je podriadený Atmel týmu a operačný systém nedovoľuje výber medzi viacerými komunikačnými zásobníkmi a protokolmi na jednotlivých vrstvách, tak ako Contiki.

LITERATÚRA

[1] ZHAO, Feng; GUIBAS, Leonidas. *Wireless Sensor Networks : An Information Processing Approach*. Boston : Morgan Kaufmann, 2004. 376 s. Dostupné taktiež z: <`http://goo.gl/T214X`>. ISBN 978-1-55860-914-3, ISBN 1-55860-914-8.

[2] MAHALIK, Nitaigour P. *Sensor Networks and Configuration : Fundamentals, Standards, Platforms, and Applications* [online]. Berlin : Springer, 2007 [cit. 2011-10-24]. Dostupné z: <http://goo.gl/11FPk>. ISBN 978-3-540-37366-7.

[3] Dominique Guinard, Vlad Trifa, Erik Wilde. *Architecting a Mashable Open World Wide Web of Things*. [online]. Technical report no. 663, February 2010 [cit. 2011-11-15]. Department of Computer Science, ETH Zurich. Dostupné z: <`http://www.vs.inf.ethz.ch/publ/papers/WoT.pdf`>.

[4] XU, Ning. A Survey of Sensor Network Applications. *IEEE Communications Magazine* [online]. 2002, 40, [cit. 2011-10-24]. Dostupný z: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.131.9647>.

[5] HOLGER, Karl; WILLIG, Andreas. *Protocols and Architectures for Wireless Sensor Networks*. Chichester (West Sussex, England): Wiley, 2007. 497 s. Dostupné z: <http://goo.gl/tzX9A>. ISBN 978-0-470-09510-2.

[6] ASÍN, Alicia. *Smart Cities sensor platform from Libelium allows to monitor noise, pollution, structural health and waste management.* Wireless sensor networks : research group [online]. June 23, 2011, no, [cit. 2011-10-22]. Dostupný z: <http://www.sensor-networks.org/index.php?page=1117417456>.

[7] ASÍN, Alicia; CALAHORRA, Manuel. *Sensor networks to monitor air pollution in cities.* Libelium : articles [online]. September 30, 2011, no, [cit. 2011-10-22]. Dostupný z: <http://www.libelium.com/smart_cities_wsn_air_pollution>.

[8] AVRAM, C., et al. Ant Routing Protocol in a ZigBee Ad Hoc Sensors Network for Radiation Level Monitoring. In: *International Conference on Automation Quality and Testing Robotics (AQTR), 2010 IEEE*. Cluj-Napoca (Romania) : [s.n.], 2010. s. 6. Dostupné z: <`http://goo.gl/9WYoT`>. ISBN 978-1-4244-6724-2, doi:10.1109/AQTR.2010.5520724 .

[9] AKYILDIZ, I. F., et al. *Wireless sensor networks: a survey.* Computer networks [online]. 2002, 38, [cit. 2011-10-22]. Dostupný z: <http://goo.gl/EK6Br>.

[10] SEONGHWAN, Cho; CHANDRAKASAN, A.P. Energy efficient protocols for low duty cycle wireless microsensor networks. In: *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2001.* Salt Lake City : [s.n.], 2001 [cit. 2011-10-24]. Dostupné z: <http://goo.gl/MpvHc>. ISBN 0-7803-7041-4, ISSN 1520-6149.

[11] BULUSU, Nirupama, et al. Scalable Coordination for Wireless Sensor Networks : Self-Configuring Localization Systems. In: *Proceedings of the 6th International Symposium on Communication Theory and Applications (ISCTA '01)* [online]. Ambleside (United Kingdom) : USC/Information Sciences Institute, 2001 [cit. 2011-10-23]. Dostupné z: <http://goo.gl/fyKMI>.

[12] HOBLOS, G.; STAROSWIECKI, M.; AITOUCHE, A. Optimal design of fault tolerant sensor networks. In: *Proceedings of the 2000 IEEE International Conference on Control Applications.* Anchorage : [s.n.], 2000. s. 467 - 472. Dostupné z: <http://goo.gl/BmHl8>. ISBN 0-7803-6562-3, doi: 10.1109/CCA.2000.897468 .

[13] PORRET, A., et al. A low-power low-voltage transceiver architecture suitable for wireless distributed sensors network. In: *IEEE International Symposium on Circuits and Systems, 2000.* Vol. 1. Geneva, Switzerland : [s.n.], 2000. s. 56-59. Dostupné z: <http://goo.gl/OHRwx>.

[14] VIEIRA, M.A.M., et al. Survey on wireless sensor network devices. In: *2003 IEEE Conference on Emerging Technologies and Factory Automation.* Vol. 1. Lisbon : [s.n.], 2003. s. 537 - 544. Dostupné z: <http://goo.gl/fkvdE>. ISBN 0-7803-7937-3, doi: 10.1109/ETFA.2003.1247753 .

[15] FAVRE, P., et al. *A 2-V 600-μA 1-GHz BiCMOS super-regenerative receiver for ISM applications.* IEEE Journal of Solid-state Circuits. Dec 1998, 12, 33, s. 2186–2196. Dostupné z: <`http://goo.gl/Io7kQ`>.

[16] LYNCH, Ciarán a Fergus O' REILLY. Processor Choice For Wireless Sensor Networks. In: *Workshop on Real-World Wireless Sensor Networks in REALWSN'05.* [s.l.] : [s.n.], 2005. s. 1–5. Dostupné z: <`http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.114.839`>.

[17] CHOI, Lynn. Power Management in Wireless Sensor Network. In: *KRnet 2006 : The 14th Korea Internet Conference* [online prezentácia]. Seoul : Korea University, Dept. of Electronics and Computer Engineering, 27. júna 2006 [cit. 2011-11-17]. Dostupné z <`http://goo.gl/yVXe9`http://goo.gl/yVXe9>.

[18] Atmel Corporation, *ATmega128* [online datasheet]. Posledná aktualizácia 8.6.2011 [cit. 15.11.2011]. Dostupné z URL: <http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf>.

[19] Texas Instruments, *MSP430F1611* [online datasheet]. Posledná aktualizácia 7.3.2011 [cit. 15.11.2011]. Dostupné z URL: <http://www.ti.com/lit/ds/symlink/msp430f1611.pdf>.

[20] CALLAWAY, E., et al. Home networking with IEEE 802.15.4: a developing standard for low-rate wireless personal area networks. *IEEE Communications Magazine*. Aug 2002, vol. 40, no. 8, s. 70-77. Dostupný tiež z: <http://goo.gl/nXY8t>. ISSN 0163-6804, doi: 10.1109/MCOM.2002.1024418 .

[21] BROWNFIELD, M., et al. Wireless Sensor Network Radio Power Management and Simulation Models. *The Open Electrical & Electronic Engineering Journal*. 2010, Volume 4., none, s. 21-31. Dostupný tiež z: <http://www.benthamscience.com/open/toeej/articles/V004/21TOEEJ.pdf>.

[22] *ZigBee Alliance* [online]. c2011 [cit. 2011-11-19]. Dostupné z: <http://www.zigbee.org/>.

[23] LEE, Jin-Shyan; HUANG, Yang-Chih. ITRI ZBnode : A ZigBee/IEEE 802.15.4 Platform for Wireless Sensor Networks. In: *2006 IEEE International Conference on Systems, Man and Cybernetics*. Volume 6. Taipei : DnE Information Service Net, 2006. s. 1462 - 1467. Dostupné z: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4274057>. doi: 10.1109/ICSMC.2006.384923 .

[24] SAJDL, Ondřej, et al. ZigBee Technology and Device Design. In: *Proceedings of the International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL'06)*. Morne (Mauritius) : [s.n.], 2006. s. 129. Dostupné z: <http://goo.gl/DvHBI>. ISBN 0-7695-2552-0, doi: 10.1109/ICNICONSMCL.2006.233 .

[25] RAN, Peng; SUN, Mao-heng; ZOU, You-min. ZigBee Routing Selection Strategy Based on Data Services and Energy-Balanced ZigBee Routing. In: *2006 IEEE Asia-Pacific Conference on Services Computing (APSCC)*. Guangzhou (Guangdong) : Applied Digital Imaging, 2006. s. 400 - 404. Dostupné z: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4041264>. ISBN 0-7695-2751-5, doi: 10.1109/APSCC.2006.116 .

[26] *HART Communication Protocol : Wireless HART Technology* [online]. HART Communication Foundation, c2011 [cit. 2011-11-21]. Dostupné z: <http://www.hartcomm.org/>.

[27] LENNVALL, Tomas; SVENSSON, Stefan; HEKLAND, Fredrik. A comparison of WirelessHART and ZigBee for industrial applications. In: *WFCS 2008. IEEE International Workshop on Factory Communication Systems, 2008.* Dresden : [s.n.], 2008. s. 85 - 88. Dostupné z: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4638746>. ISBN 978-1-4244-2349-1, doi: 10.1109/WFCS.2008.4638746 .

[28] KIM, Anna N., et al. When HART goes wireless: Understanding and implementing the WirelessHART standard. In: *Proceedings of the 13th IEEE International Conference on Emerging Technologies and Factory Automation.* Hamburg : [s.n.], 2008. s. 899 - 907. Dostupné z: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4638503>. ISBN 978-1-4244-1505-2, doi: 10.1109/ETFA.2008.4638503.

[29] HUI, Jonathan W.; CULLER, David E. Extending IP to Low-Power, Wireless Personal Area Networks. *IEEE Internet Computing.* July-Aug. 2008, vol. 12, no. 4, s. 37-45. Dostupný tiež z: <http://www.cs.berkeley.edu/~jwhui/pubs/jhui-ic-6lowpan.pdf>. ISSN 1089-7801.

[30] MEMSIC, *IRIS* [online datasheet]. Posledná aktualizácia 16.9.2011 [cit. 3.12.2011]. Dostupné z URL: <http://goo.gl/P0pUE>.

[31] MORÁVEK, Patrik, Dan KOMOSNÝ, Milan ŠIMEK a Ľubomír MRÁZ. Energy demands of 802.15.4/ZigBee communication with IRIS sensor motes. In: *34th International Conference on Telecommunications and Signal Processing (TSP).* Budapest, 2011, 69 - 73. ISBN 978-1-4577-1410-8. DOI: 10.1109/TSP.2011.6043770. Dostupné z URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6043770

[32] GP Batteries, *GP250AAHC.* Dostupné z URL: <http://www.gpbatteries.com/pic/GP250AAHC-r1.pdf>.

[33] *TinyOS Documentation Wiki* [online]. 2009, 1 August 2011 [cit. 2011-12-07]. Dostupné z: <http://docs.tinyos.net/tinywiki/index.php/Main_Page>.

[34] Atmel Corporation, *AT86RF230* [online datasheet]. Posledná aktualizácia 15.4.2010 [cit. 3.12.2011]. Dostupné z URL: <http://www.atmel.com/dyn/resources/prod_documents/doc5131.pdf>.

[35] Atmel Corporation, *ATmega1281* [online datasheet]. Posledná aktualizácia 24.5.2011 [cit. 3.12.2011]. Dostupné z URL: <http://www.atmel.com/dyn/resources/prod_documents/doc2549.pdf>.

[36] *The Contiki Wiki* [online]. 2010, 29 February 2012 [cit. 2012-03-04]. Dostupné z: <http://www.sics.se/contiki/wiki/index.php/Main_Page>

[37] DUNKELS, Adam, Björn GRÖNVALL a Thiemo VOIGT. Contiki: a lightweight and flexible operating system for tiny networked sensors. In: *Proceedings of the First IEEE Workshop on Embedded Networked Sensors (Emnets-I)* [online]. Tampa, 2004 [cit. 2012-05-23]. Dostupné z: <http://www.sics.se/~adam/dunkels04contiki.pdf>.

[38] Protothreads. *Contiki 2.5* [online]. April 2011 [cit. 2012-03-04]. Dostupné z: <http://dak664.github.com/contiki-doxygen/>.

[39] ATMEL. *BitCloud Developers Guide*. San Jose, 2011.

ZOZNAM SYMBOLOV, VELIČÍN A SKRATIEK

ACK Potvrdenie – Acknowledgment

AČ Analógovo-číslicový

AODV Smerovanie s vektorom vzdialenosti na vyžiadanie v Ad hoc sieťach – Ad hoc On-Demand Distance Vector Routing

APS Podvrstva podpory aplikácií – Application support Sublayer

ARQ Automatická požiadavka opakovania prenosu – Automatic Repeat reQuest

BSP Balíček podpory dosky platformy – Board Support Package

CCA Hodnotenie voľného kanálu – Clear Channel Assesment

CSMA/CA Viacnásobný prístup s načúvaním nosného kmitočtu s predchádzaním kolízie – Carrier Sense Multiple Access with Collision Avoidance

CTR Rozhodujúci dosah prenosu – Critical transmitting range

CTS Pripravený na posielanie – Clear to Send

DPM Dynamická správa napájania – Dynamic Power Management

DSSS Technika priameho rozprestreného spektra – Direct Sequence Spread Spectrum

EUI Rozšírený unikátny identifikátor – Extended Unique Identifier

FFD Zariadenie s plnou funkcionalitou – Full Function Device

FHSS Technika rozprestreného spektra so zmenou nosnej – Frequency Hopping Spread Spectrum

GTS Garantované časové sloty – Guaranteed time slots

IEEE Inštitút pre elektrotechnické a elektronické inžinierstvo – Institute of Electrical and Electronics Engineers

IETF Internet Engineering Task Force

ISM Frekvenčné pásma pre priemyselné, vedecké a medicínske účely – Industrial, Scientific and Medical radio bands

LLC Riadenie logického spoja – Logical Link Control

LQI Indikácia kvality spojenia – Link Quality Indication

LR-WPAN Bezdrôtová personálna sieť s nízkou mierou prenosu – Low-Rate Wireless Personal Area Networks

MAC Riadenie prístupu k médiu – Medium access control

MCU mikrokontrolér – Micro Controlling Unit

MTU Maximálna prenosová jednotka – Maximum Transmission Unit

ND Neighbor discovery

OUI Unikátny identifikátor organizácie – Organization Unique Identifier

PAN Osobná sieť – Personal Area Network

PHY Fyzická vrstva – Physical layer

PPDU Dátová jednotka protokolu fyzickej vrstvy – PHY protocol data unit

RFD Zariadenie s obmedzenou funkcionalitou – Reduced function device

RTS Žiadosť o posielanie – Request to Send

TC Riadenie topológie – Topology control

TDMA Viacnásobný prístup s časovým delením – Time Division Multiple Access

UHF Rozsah veľmi vysokých frekvencií – Ultra High Frequency

WSN Bezdrôtová senzorová sieť – Wireless sensor network

ZDO Objekt ZigBee zariadenia – ZigBee Device Object

ZOZNAM PRÍLOH

A ŠTRUKTÚRA MODULU OSCIC.NC

```
#include "Timer.h"
#include "Osci.h"
module OsciC @safe()
       {
         uses {
           interface Boot;
           interface SplitControl as RadioControl;
           interface AMSend;
           interface Timer<TMilli>;
           interface Read<uint_16_t> as Voltage;
           interface LowPowerListening as LPL;
           }
       }
implementation
{ message_t sendBuf; bool sendBusy;
       ...
task void readTask();
task void sendTask();
event void Boot.booted() {
       LPL.setLocalWakeupInterval(LPL_INTERVAL);
       local.interval = DEFAULT_INTERVAL;
       local.id = TOS_NODE_ID;
       if (call RadioControl.start() != SUCCESS)
              report_problem();
         }
void read(){...}
void send(){...}
task void readTask() { read(); }
task void sendTask() { send(); }
void startTimer() {
           call Timer.startPeriodic(local.interval);
           reading = 0;
         }
event void RadioControl.startDone(error_t error) {
           startTimer();
         }
         ...
event void Timer.fired(){...}
}
```

B LOWPOWERLISTENING A MCUSLEEPC

```
interface LowPowerListening {
command void setLocalWakeupInterval(uint16_t intervalMs);
command uint16_t getLocalWakeupInterval();
command void setRemoteWakeupInterval(message_t *msg, uint16_t ms);
command uint16_t getRemoteWakeupInterval(message_t *msg);
}
```

```
module McuSleepC @safe() {
provides {
           interface McuSleep;
           interface McuPowerState;
       }
uses {
           interface McuPowerOverride;
       }
}
implementation {
 ...
async command void McuSleep.sleep() {
           uint8_t powerState;
         powerState = mcombine(getPowerState(),
           call McuPowerOverride.lowestState());
SMCR=(SMCR & 0xf0) | 1 << SE | read_uint8_t(&atm128PowerBits[powerState]);
         sei();
         asm volatile ("sleep" : : : "memory");
         cli();
     CLR_BIT(SMCR, SE);
         }
        ...
default async command mcu_power_t McuPowerOverride.lowestState() {
           return ATM128_POWER_DOWN;
         }
}
```

C TASKHANDLER ENDDEVICE

```
static void taskHandler(void)
{
  switch (endDeviceState)// Stavy definované štruktúrou endDeviceState
  {
    case END_DEVICE_STATE_START_MEASURE:// Zahájenie merania zo senzorov
      typeChanged = false;
      appMessageBuffer.msg.report.type = currentType;
      endDeviceState = END_DEVICE_STATE_MEASURING;
      boardAbstractionReadSensor(1, sensorHandler);
//Čítanie zo senzoru, hodnota sa vracia návratovou funkciou sensorHandler
      break;

    case END_DEVICE_STATE_MESSAGE_SENDING:// Posielanie správy
      if (TRANSMISSION_STATE_IDLE == transmissionState)
// Kontrola poslanie predošlých dát
      {
        transmissionState = TRANSMISSION_STATE_BUSY;
        APS_DataReq(&apsDataReq);
        boardAbstractionShowIndication(APP_INDICATION_DATA_TRANSMITED);
// LED indikácia poslania správy
      }
      break;

    case END_DEVICE_STATE_PERIPHERY_CLOSING:
      if (boardAbstrationIsIdle())// Prechod periférii do nečinného stavu
      { // Príprava spánkového režimu
        endDeviceState = END_DEVICE_STATE_SLEEP_PREPARE;
        boardAbstractionClose();
// Vypnutie hardvérových periférií LED, senzory
      }
// Odloženie úlohy pre čakanie na dokončenie operácii na BSP vrstve
      SYS_PostTask(APL_TASK_ID);
      break;

    case END_DEVICE_STATE_SLEEP_PREPARE:
      prepareToSleep();
// Funkcia pre vyžiadanie prechodu do spánkového režimu
      break;
```

```
    case END_DEVICE_STATE_AWAKENING:// Stav prebudenia
      zdoWakeUpReq.ZDO_WakeUpConf = ZDO_WakeUpConf;
// ZDO WakeUp obsluhovač potvrdenia
      ZDO_WakeUpReq(&zdoWakeUpReq);
// ZDO WakeUp zasielanie požiadavku prebudenia
      break;

    case END_DEVICE_STATE_MEASURING: // čaká navrátové volanie od senzoru
    case END_DEVICE_STATE_SLEEP:
    case END_DEVICE_STATE_WAIT_JOIN:
// čaká notifikáciu od nadradeného automatu z lowpower.c
      break;
  }
}
```